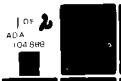
AD-A104 888 BATTELLE COLUMBUS LABS OH F/G 9/2
AN IPSS-BASED MODEL-BUILDING METHODOLOGY FOR RANKING AND EVALUA--ETC(U)
OCT 79 J D BROWNSMITH, J S CARSON

UNCLASSIFIED NL



LEVEL II

(15)

AIRMICS FINAL REPORT*

An IPSS-Based Model-Building Methodology for Ranking and Evaluating Computer Hardware/Software Systems

By

Joseph D. Brownsmith, Ph.D.
Database Systems Research and Development Center
Department of Computer and Information Sciences
University of Florida
Gainesville, Florida 32611

John S. Carson II, Ph.D.
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

William H. Hochstettler III, M.S.
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

15 October 1979

* This research was performed for AIRMICS through the United States Army Research Office with the financial support of the Battelle Scientific Services Program-D.O. 1271

AIRMICS FINAL REPORT*

An IPSS-Based Model-Building Methodology for Ranking and Evaluating Computer Hardware/Software Systems

By

Joseph D. Brownsmith
Database Systems Research and Development Center
Department of Computer and Information Sciences
University of Florida
Gainesville, Florida 32611

John S. Carson II, Ph.D.
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

William H. Hochstettler III, M.S.
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

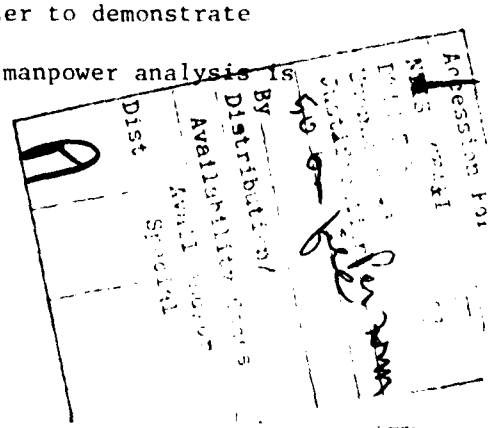
15 Oct 79

*This research was performed for AIRMICS through the United States Army Research Office with the financial support of the Battelle Scientific Services Program - D.O. 1271.

Executive Summary

The objective of this research was to design and implement a model building methodology for simulating U.S. Army computer hardware/software systems. Computer systems are characterized in terms of file parameters, hardware specification, and software use of files. These descriptions reside in a model library and are the building blocks in the model synthesis process. The Information Processing System Simulator (IPSS) language was used to encode these descriptions and to represent the sequence of computer activities for application program processing (e.g., job scheduling, buffer management, channel program).

Two computer systems were compared using this methodology. Simulation models were written for an IBM 360 Model 30 computer and a Honeywell Level 6 minicomputer. A subset of the U.S. Army Standard Installation/Division Personnel System (SIDPERS) provided a common loading for both systems. Data was collected on an operational IBM 360/30 and the IPSS model was validated. The statistical results, derived from IPSS, indicate resource utilization (for both hardware and software resources), elapsed time, and queueing. Our results project that an eight hour execution of SIDPERS on the IBM 360/30 would execute in approximately two and one-half hours on the Honeywell machine. Models of several hardware variations were prepared in order to demonstrate responsiveness capabilities of the methodology. A manpower analysis is provided for guidance in estimating future work.



ACKNOWLEGMENTS

The Authors would like to acknowledgment the able project administration and support of Mr. James Gantt and Mr. Allan Curry of AIRMICS. We also thank CPT Paul A. White and 1LT Robert H. Fleming of Quality Assurance, USACSC, for providing data and other information, and for making valuable suggestions on the content and presentation of this final report. In addition, MAJ Schuler, SP5 Sweatman, and SGT Huerd of the Fort Stewart Division Data Center provided us with much valuable operating data and SIDPERS data, and allowed us to observe a SIDPERS run. Mr. Dixie Tompkins, of the Management Information Systems Office, FORSCOM, Fort MacPherson, also provided us with data, manuals and other valuable information about SIDPERS software and files. Finally, we deeply appreciate the work and dedication of Mr. Bruce E. Wilhite, an undergraduate student at the University of Florida, who installed and maintained IPSS and provided computer support during the project.

TABLE OF CONTENTS

Table of Contents continued.

LIST OF TABLES

LIST OF FIGURES

List of Figures Continued.

1. INTRODUCTION

This report is in response to the Laboratory Research Cooperative Program Statement of Work TCN: 79-245, which required the services of three research scientists on a short-term project to develop simulation models of computer systems. The objective of this research was to produce a model building methodology using the Information Processing System Simulator (IPSS) to develop a ranking and evaluation procedure for computer hardware/software systems. Five specific tasks were identified:

1. Using IPSS, specify, design, build, test, validate, verify and document a model of an existing Army computer hardware/software system (such as the U.S. Army Base Operations System (BASOPS) implemented on IBM 360/40 equipment).
2. Using IPSS, specify, design, build, test, validate, verify and document a model of an advanced Army computer hardware/software system (such as a minicomputer data base oriented system).
3. Specify and collect data needed to build the models of computer hardware/software systems specified in 1 and 2 above.
4. Develop measures to allow for the ranking and evaluation of computer hardware/software systems. Factors should include, but not be limited to, growth rate, workload, software, variance in configurations, and new applications.
5. Arrange for and provide computer support services, to include computer time, disk storage, and IPSS software support.

This final report to AIRMICS reflects the background activity, purpose, procedures, documentation, and summary of work performed under each of the above tasks.

Tasks 1, 2, 3 and 5 have been completed in full; we could not complete task 4 due to lack of time. As the Army is considering replacement of certain of its computer systems, we did consider, relative to task 4, measures for evaluating computer systems when the major factor is variance in hardware configurations.

1.1 SUMMARY OF RESEARCH ACTIVITIES

The primary objective of this project was accomplished by designing, building, testing, verifying, and validating two basic models of Army computer systems. The first model was of an existing Army computer system, namely, an IBM 360 Model 30 with a selected subset of the Standard Installation Division Personnel System (SIDPERS) basic cylce for loading. This model provided a frame of reference and was validated. The second model was of an advanced computer system (one not currently operational) that was considered to be typical of potential Army purchases. This system was a Honeywell Series 60 Level 6 Model 47 minicomputer with the same SIDPERS basic cycle for loading. Several variations on the basic hardware architecture were modeled and analyzed.

These models were compared against the same workload, the first four jobsteps of SIDPERS. The results of such comparisons allow for a relative ranking of the various systems. Such a ranking is the first step towards determining what computer will meet the needs of the location

being examined. The IPSS approach is unique in that almost all currently available simulation techniques deal only with representative batch oriented systems while IPSS has special facilities which will allow the modeling of advanced computer features such as data bases, networks, and interactivity.

Of primary concern is the acceptance of the modeling methodology within the Army. Thus we have concentrated on validating a model of a simple and typical hardware/software system. The underlying assumption is that credibility will transfer to models of more advanced systems given a well validated basic model.

At the start of the project, AIRMICS personnel provided us assistance in selecting the computer systems that we were to model. After a preliminary investigation of the type of processing SIDPERS performs and the hardware configurations, we proceeded as follows:

1. Developed the IPSS Application Processing System (IAPS) methodology for representing application systems processing.
2. Implemented the methodology in IPSS.
3. Collected data on four SIDPERS job steps and the two computer hardware configurations.
4. Coded the hardware and software descriptions for the above in IPSS.
5. Verified the IPSS models.
6. Validated the model of the IBM 360/30.
7. Performed experiments using alternate hardware configurations.
8. Analyzed the results.

1.2 SUMMARY OF RESULTS

This section summarizes the major contribution of our research project. These results are discussed in detail in the sections referenced.

First, we demonstrated the appropriateness of using IPSS for modeling typical U.S. Army computer hardware/software systems, and showed that the simulation technique can provide data useful for the comparison of alternative computer systems. To ease the task of modeling in IPSS, we provided a high level modeling approach through our IPSS Application Processing System (IAPS) methodology which is able to accommodate any level of detail desired by the simulation user. (See Chapter 3)

In conjunction with IAPS, we established a basic library of model components which allows a user to easily and quickly build a model of a large number of design alternatives. This library can be modified and the number of its members increased so as to enhance future Army modeling needs. The library as it currently exists is described in Appendix E.

We identified the types of verification and validation data needed and their sources within the U.S. Army Computer Systems Command. (See Chapter 4) The ready availability of needed data would greatly shorten the time needed to complete any future modeling efforts.

We demonstrated the feasibility of the IAPS methodology, and the usefulness of the data collected by modeling and comparing several design alternatives for the Army CS_3 hardware/software system. This

effort included validation of a model of an existing system, development of models of nine alternative hardware configurations, and a comparison of the different systems. (See Section 5.2 for the hardware alternatives, Sections 4 and 5 for the SIDPERS model, and Section 7 for the simulation results.) In addition, we have given a manpower analysis of the current project and several possible future projects. (See Section 8.)

We identified appropriate measures for the comparison and ranking of production, batch oriented Army computer systems. Those measures which can be estimated via simulation, and which are collected automatically by IPSS, include job elapsed time, resource utilization, and queuing statistics. (See Section 7.)

All in all, we believe that the complete IAPS simulation methodology is a feasible and potentially useful approach in the Army's evaluation and comparison of alternative computer systems.

The remainder of this report is organized as follows. In Section 2, we discuss the background to the project and the motivation for our modeling approach. The methodology for modeling and evaluating computer hardware/software systems is presented in Section 3. Sections 2, 4, and 5 present the specific problems of modeling SIDPERS and the selected computer hardware architectures. Section 6 identifies the structure of our IPSS model while Section 7 presents the results of our modeling experiments. A summary of the time required for various modeling activities is given in Section 8. We finish with a summary, recommendations and conclusions in Section 9. A number of Appendices contain auxiliary material.

2. BACKGROUND AND APPROACH TO COMPUTER EVALUATION

2.1 BACKGROUND TO THE PROJECT

The United States Army is about to enter a period in which several large purchases of computer hardware/software systems are to be made. For example, one purchase could involve the replacement of over 40 computer installations. Choosing one machine to work at over 40 places would be complex enough, but in this case theoretically there could be over 40 different machines chosen. A computer vendor can bid on any number of sites with any combination of equipment. The workload profile at the locations for the machines is radically different. A minicomputer might work very well at one place while another must have a large main-frame.

Currently the sites have IBM 360/30's, IBM 360/40's, and IBM 360/50's, some with single peripherals, some with dual peripherals, some running the DOS operating system (or the enhanced DOS system DOS-E) and some with OS. A major portion of the software is consistent from location to location, but the volume of transactions is drastically different. All current work is batch oriented.

The Army desires to purchase new equipment to replace these machines. They want to add interactive capabilities while retaining batch processing for some applications. One requirement of the new computers is that they process the current work in one eight hour shift five days a week. (Currently most sites are running 24 hours a day five to seven days a week).

With recent technological advances, selecting even one computer is almost beyond human capability if one is to easily and fairly compare all of the machines that vendors would contend can do a given job. Current methods, such as benchmarking, fall short of solving the problem. Simulation appears to be a very attractive approach because of its flexibility and power in representing complex activities. Thus, this project requires the use of discrete event digital simulation to assist in the selection and evaluation of computer hardware/software systems.

This project specifically required the use of the Information Processing System Simulator (IPSS) to rank and evaluate computer hardware/software systems. IPSS is a special-purpose discrete event digital simulator system which was specifically designed to facilitate the investigation of the behavior of complex computer-based information processing systems (DEL77, DEL78a).

One significant feature of IPSS is its ability to characterize a computer's I/O subsystem. The IPSS language contains a rich set of instructions for describing control units, channels, disk and tape drives, and unit record equipment. The IPSS "service" concept permits a flexible characterization of the acquisition, use and release of the secondary storage "facilities".

These features are important because of the Army's predominately I/O oriented computer systems. Thus, a detailed modeling of the I/O subsystem should be of great potential value in identifying bottlenecks and effects of hardware/software changes. IPSS provides the capacity for detailed modeling of this computer subsystem and

also automatically collects elapsed time, resource utilization and queueing statistics for the user.

Although IPSS is a prototype system, it proved to be an able tool for characterizing salient features of SIDPERS application as well as the hardware characteristics of the IBM Model 30 and the Honeywell Level 6 minicomputer. An overview of the IPSS methodology is provided in Appendix A.

2.2 APPROACH TO MODELING AND VALIDATION

The completion of the specific research tasks outlined in Section 1 required the resolution of two basic issues, namely:

1. How to represent application program software in a simulation model, and
2. What data was available within the U.S. Army for validation of simulation models of computer hardware/software systems.

Since our resolution of these tasks was both time consuming and crucial to the results of the project, an overview of our approach is given here.

The first task, how to model computer software, can be rephrased as: What is the best approach for modeling large software systems by using IPSS. (IPSS has considerable flexibility and power in representing computer hardware, so that aspect of computer system modeling was not a problem). Software systems such as SIDPERS contain many large COBOL programs. We recognized that a detailed statement-by-statement or even paragraph-by-paragraph description of the processing logic

would be far too time consuming for this project. IPSS is, however, capable of representing processing logic at this level of detail and this approach may prove to be valuable for some programs, such as operating system routines, or for more refined results. Even if this approach were used, only one or two simple programs could be characterized in the time allowed, giving little insight into the processing characteristics of a large system. Clearly, we were challenged to produce a faster modeling approach which would both retain a useful level of accuracy of the detailed approach and provide flexibility for the modeler.

SIDPERS, the software system selected, has been modeled before using the software simulation package CASE (ADL75, SWE76). In addition, an IPSS model of several SIDPERS programs was part of an IPSS SIDPERS/IDMS simulation study (BRO77, DEL78), and a DIMUI model also simulated these programs (SCH77). The methods for representing software processing in these models were studied but not adopted in our methodology. The CASE approach represents files and file processing in an easily-understood and consistent manner, but overall was not judged to be a suitable methodology because of its lack of flexibility (for an evaluation of CASE versus IPSS see ROS78). The previous IPSS and DIMUI approaches were rejected since they modeled interactive SIDPERS programs by representing every call to a DBMS routine. (The batch SIDPERS programs that we modeled requested I/O to many types of files in the absence of a DBMS).

Our approach was to detail I/O processing on a file by file basis, and, in less detail, to characterize program CPU loading. This is consistent with the IPSS methodology and also allows the

modeler freedom to change the procedural structure of the model. Our methodology, called IAPS (IPSS Application Processing System), is reported in Section 3.

The second task we faced was the selection of a hardware/software system for which validation data existed. Validation is the process of determining the degree of validity of a simulation model. A valid model is one which is capable of accurately measuring, predicting and representing a system. The validation process proceeds to build an acceptable level of confidence that an inference about a simulated process is a correct or valid inference for the actual process. Validation of a model is performed by a comparison of the recorded observations of the real process with simulator outputs from a verified model, thereby establishing the versimilitude of the model and the real world process (MIH76a, MIH76b). Seldom, if ever, will validation result in a "proof" that the model is a correct or "true" representation of the real process (VAN69). Verification, on the other hand, is the comparison of the model's responses with those anticipated if the model's structure were programmed as intended (MIH76b). This means testing the outputs of the random number generators as well as checking that the computer program correctly executes the logic desired by the modeler.

With assistance of USACSC personnel we decided, early in the project, to model a subset of SIDPERS executing on the IBM Model 30 utilized at the Division of the Army's organization and to compare the results against a Honeywell Level 6 Model 47 minicomputer using the same SIDPERS workload. We also determined that GRASP step accounting data was the most important requirement for a detailed validation of our simulation models. Detailed validation of a SIDPERS job step requires the following:

1. GRASP step accounting data
2. The operator console's log (for the job step)
3. SYSLIST (for the job step)
4. Listing of specified data sets
5. VTOC listing of all disk packs which were on-line during the job step
6. Researchers present in the machine room during execution of the job step.

Since GRASP step accounting data was not available on a 360/30 but was available on a 360/50, we considered the following alternatives:

A. Model the 360/50, and validate the model
B. Model the 360/30, which couldn't be validated in detail
C. Do both A and B

The first alternative was rejected since it would not permit the comparison desired by the Army between the IBM and Honeywell computers. The advantage to alternative A was that we would produce a model which could be validated in detail, thus demonstrating the effectiveness of our methodology. We did not have the time to produce three models so alternative C was also rejected. Instead, we concentrated on getting as much data as possible from a SIDPERS cycle running on a 360/30.

Section 4 details our data collection activities for the SIDPERS Basic Cycle. The next section presents our approach to modeling hardware/software systems for performance evaluation, ranking and selection.

3. THE IPSS APPLICATION PROCESSING SYSTEM METHODOLOGY (IAPS)

3.1 THE IAPS MODELING PERSPECTIVE

The problem addressed in this reserach is the design and implementation of a model building methodology to assist in the evaluation of computer hardware/software systems. The goal is a methodology with the widest possible applicability to the user community. Therefore, the IPSS design goals have been adopted, namely:

1. Breadth of Applicability -- the ability to model the behavior of contemporary and forseeable system architectures and operating environments;
2. Functional View of Systems -- the ability to identify and characterize system components and activities based on their function, independent of a particular architecture or environment;
3. Top Down, Modular Model Synthesis -- the ability to model to a level of detail commensurate with research objectives;
4. Expandable Structure -- the capability to incorporate new, higher level descriptive facilities and performance measures into the methodology and simulation system; and
5. Flexibility of Use -- the ability to be used by a wide spectrum of modelers from the experienced system analyst/designer and researcher to the practitioner and student.

Because of the wide range of knowledge required for modeling computer systems, the IAPS methodology reported in this research distinguishes four distinct modeling functions and provides facilities and tools for each. These functions partition the modeling and evaluation of computer hardware/software systems into a set of activities to be performed by:

1. the User,
2. the Modeler,
3. the Simulator, and
4. the IPSS Analyst.

These activities are summarized in Figure 3-1. As shown, the Modeler is responsible for the creation and maintenance of model libraries, the User for the selection of *library members to synthesize a model*, the Simulator and IPSS Analyst for maintaining IPSS source code and execution facilities. We now define these job functions in more detail.

User - that person or persons whose responsibility is the evaluation of computer hardware/software systems. The user conducts modeling experiments by selecting pre-defined model components from a model library, and selects execution options. The user validates the model, analyzes the simulation results and performs the required evaluation.

Modeler - that person or persons whose primary concern is with the application system to be modeled and with the hardware environment on which it will execute. The modeler builds and maintains

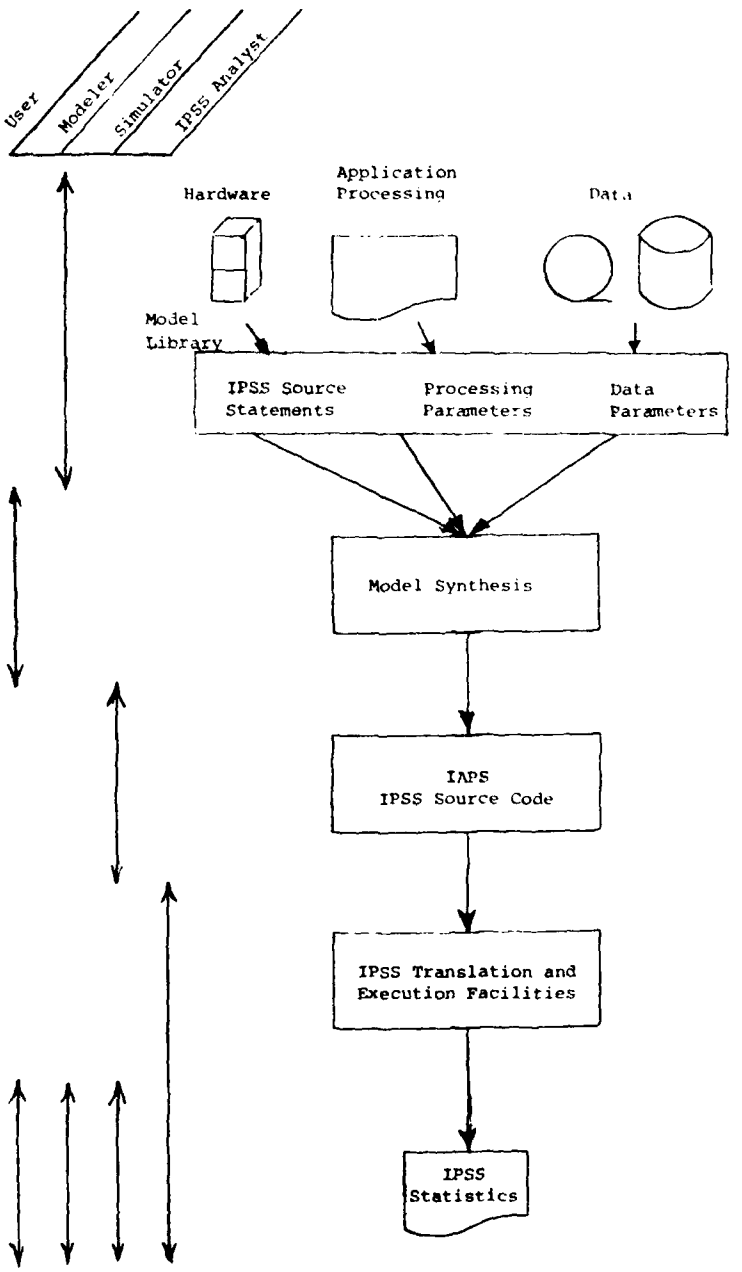


Figure 3-1. Overview of the IAPS Methodology

the model library of software and hardware components. The modeler is not concerned with the structure or execution of the IPSS model, but is concerned with model verification.

Simulator - that person or persons whose primary concern is with the structure and execution of the IAPS model. The Simulator codes user-required special-purpose IPSS routines, incorporates these routines into the model, and verifies their correctness.

IPSS Analyst - that person or persons who have a detailed knowledge of the inner workings of the IPSS simulator. This includes the source language translation process, the simulation driver, facility definitions and tables.

User level activities were established so that model synthesis and experimentation could be easily accomplished. Hardware characterization and workloads can be changed by the User without any change to the IPSS model itself. This approach assumes a library of computer system characterizations. Our research is the first step in providing an IPSS system library for the User.

The role of the User is distinct from that of the Modeler, Simulator and IPSS Analyst; the major distinction is that the User produces no IPSS source code or workload characterizations. The Modeler and Simulator may be the same person or persons. They must coordinate their activities so that the resulting simulation can be validated. For example, the Modeler describes computer hardware using IPSS statements, but it is the Simulator who describes the sequences of the IPSS simulator's acquisition, use and release of this hardware.

The Modeler, Simulator and IPSS Analyst each share a common set of modeling activities. As summarized in Table 3-1, these activities are: Hardware characterization, software description, sequence of activities, data description and model verification. As shown in the Table, the Modeler's role is independent of any procedure oriented code. The Simulator has the responsibility for maintaining the IAPS source code, and relies on the IPSS Analyst for special functions or requirements.

The remainder of this section is organized as follows. Section 3.2 outlines the User activities, Section 3.3 presents the Modeler view, Section 3.4 discusses the Simulator view and relates it to the Modeler. The IPSS Analyst function is presented in Section 3.5.

3.2 THE IAPS MODELING APPROACH: THE USER'S ROLE

The user is defined to be that person or persons with overall computer system evaluation responsibility for a given project. As shown in Figure 3-2, the user accepts and clarifies a set of evaluation requirements and produces evaluation documentation through:

- o interaction with the Modeler function,
- o interactive model synthesis,
- o validation of model results, and
- o analysis and evaluation of computer hardware/software systems.

The User interacts with the Modeler in order to ensure that the desired model library members are present for the model synthesis phase. The Modeler may be required to change the existing library members, add new ones, or to add capabilities to the IPSS model itself (such as DBMS processing) in order to satisfy the User's modeling requirements.

Table 3-1. Modeling Activities

Modeling Activity	Modeling Perspective		
	Modeler (Section 3.3)	Simulator (Section 3.4)	IPSS Analyst (Section 3.5)
1. Hardware Characterization	IPSS Hardware-oriented facilities	Channel program service	Facility table definitions
2. Software description	Execution group statement	IPSS endo services	IPSS statement parsers
3. Sequence of activities	Ordering of execution group statement	IPSS services, Equates	IPSS driver, Current and future event queues
4. Data description	User and system file tables	IPSS data base component	Logical record to physical address translation
5. Verification	Queuing and Utilization of hardware facilities	Sequence of events in the model	Random number generator, etc.

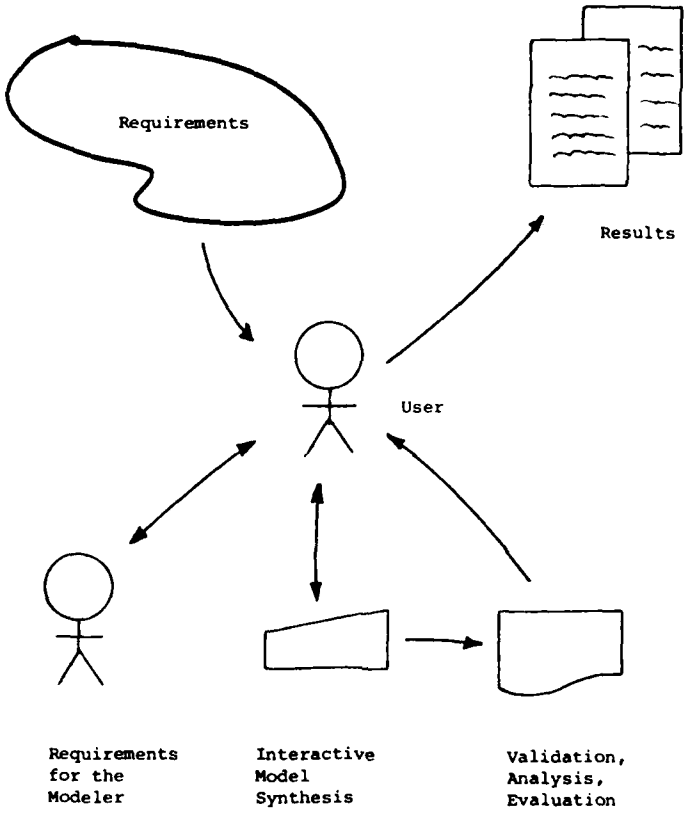


Figure 3-2. User's Role in the IAPS Methodology

The next step in the User process is interactive model synthesis. This produces an execution-ready model through the selection of a priori defined model components from the model library. This process is illustrated in Figure 3-3. We have implemented interactive model synthesis, and used it to produce the results reported in this research. An example of the User-computer interaction sequence is presented in Figure 3-4.

We designed, but did not implement (due to lack of time) a more elaborate model synthesis procedure which would allow the user to modify some of the existing library members in order to tailor them for specific processing needs. As shown in Figure 3-5, we envision that the software processing and data base description members of the model library could be so tailored. This would require more user interaction than now required but would enhance flexibility. This approach is further discussed in Section 9.

3.3 THE IAPS MODELER VIEW OF COMPUTER HARDWARE/SOFTWARE SYSTEMS

Figure 3-6 shows that the Modeler's responsibilities include the preparation of data for input to the model, and the verification of resultant simulation statistics. Input data preparation involves the following:

1. Description of the system hardware,
2. Description of the applicaiton processing workload
3. Description of the characteristics of the data files used by the application, and
4. Representing these descriptions according to the IAPS methodology specifications and storing them in the IAPS model library.

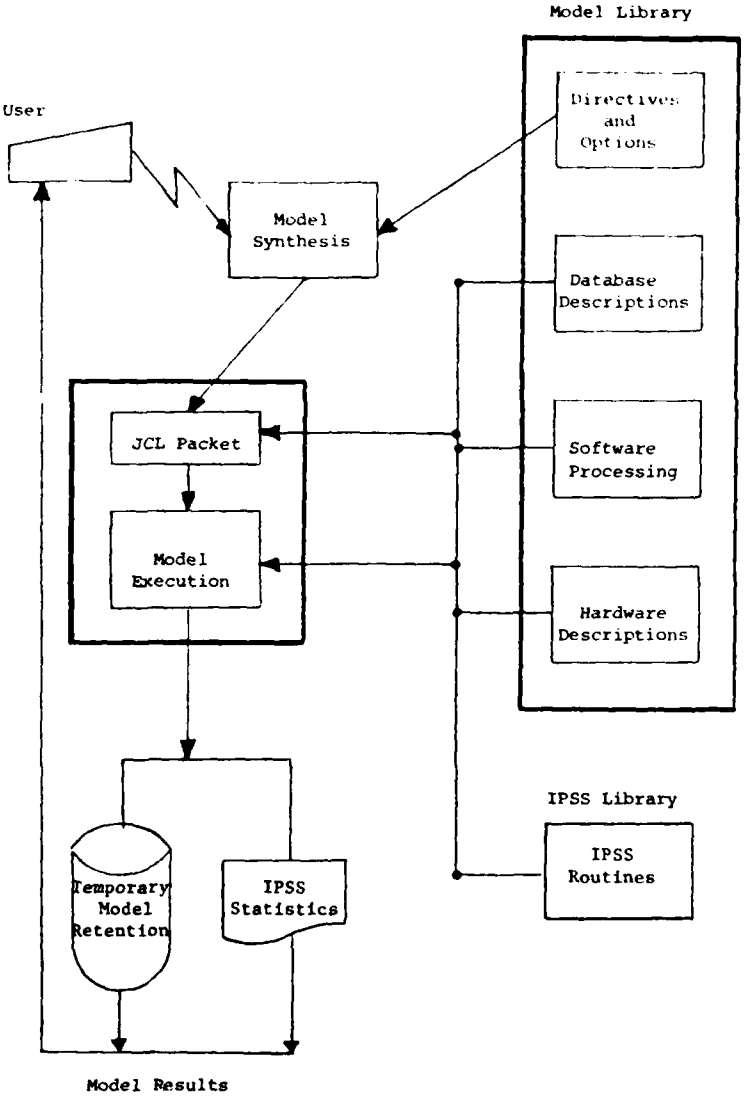


Figure 3-3. User's View of IAPS Model Synthesis and Execution

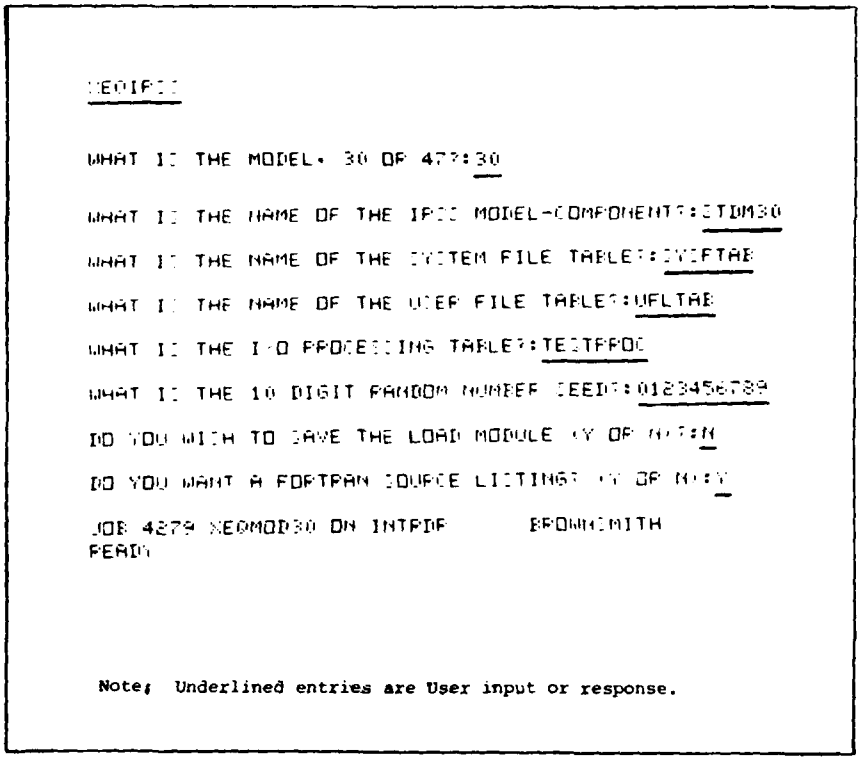


Figure 3-4. Example of Interactive Model Synthesis

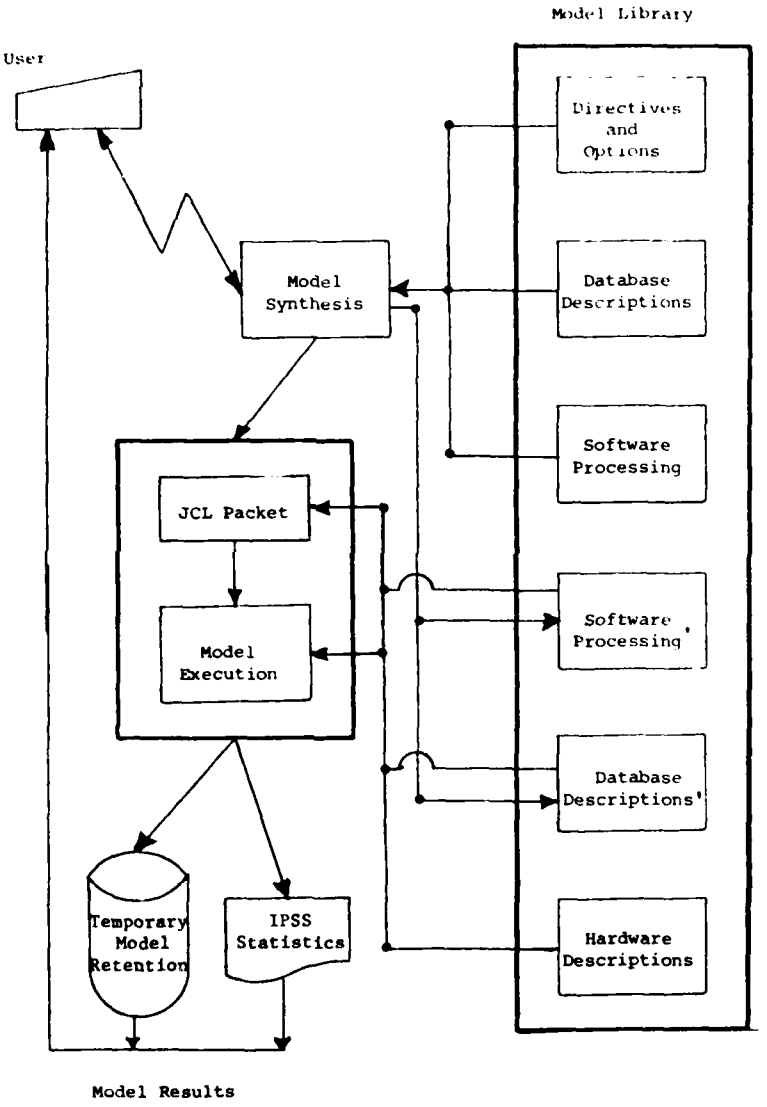


Figure 3-5. User's View of IAPS - Proposed

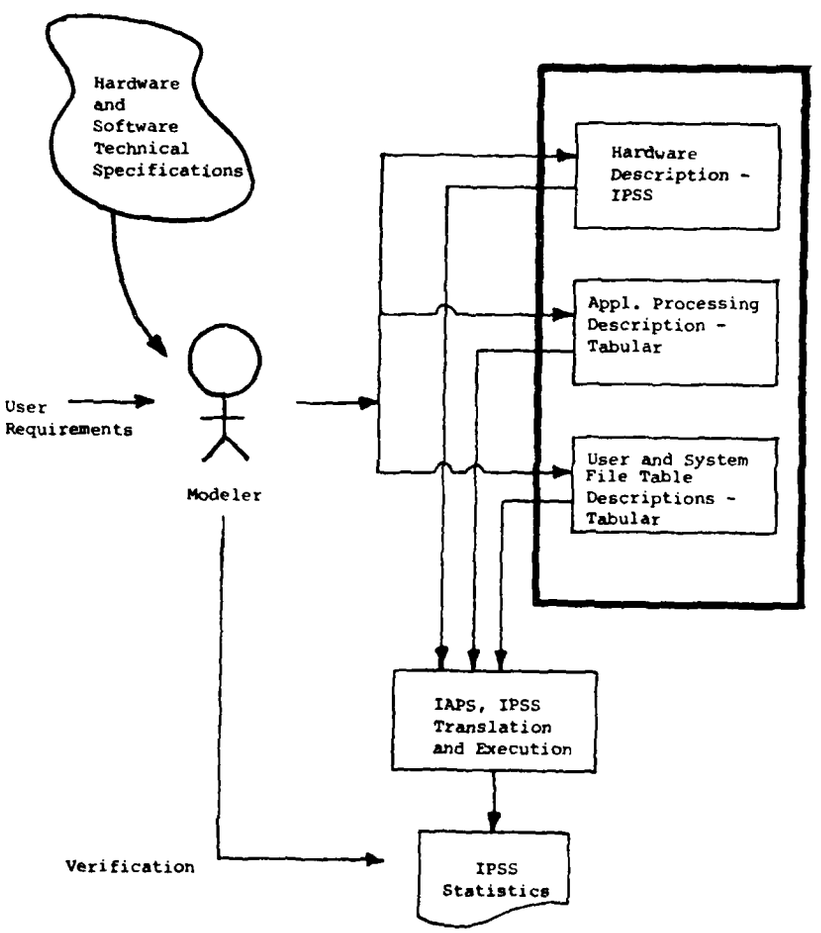


Figure 3-6. Modeler's Role in the IAPS Methodology

An overview of these Modeler activities is now presented. Details are found in the Appendices as noted in the text.

Description of System Hardware

The Modeler is responsible for identifying the basic types of computer devices for the system being modeled. As shown in Table 3-2, the devices primarily reflect the computer's mainframe and secondary storage subsystem. For each device identified, the modeler provides a detailed functional specification which indicates capacity, speed, and special features. A list of the type of data collected for disk, drum, tape, and unit record devices is given in Table 3-3. This type of data is usually readily available in vendor's technical system reference manuals. The Modeler then encodes this data into IPSS statements in a straight-forward way. Examples of these IPSS statements are found in Appendix B.

Description of Application Processing Characteristics

The application workload and its data files are characterized by two types of tables which are prepared by the modeler. These tables are called the Application File Table (AF Table), and the Application Processing Table (AP Table). The Application File Table gives detailed information about the files being processed from the application program point of view. The Application Processing Table gives, in outline fashion, a step by step description of application processing.

The Application File Table (AF Table)

The AF Table describes the characteristics of files as known by the application program. Each entry in this table describes a single file and contains: a file-identifier, the logical record length and

Table 3-2. Modeler Checklist for Computer System Hardware

Device Type
CPU
Main memory, cache
Channels (multiplexor, selector)
Disk Units, Disk Controller
Tape Units, Tape Controller
Drum Units, Drum Controller
Operator's Console
Line Printer
Card Reader
Card Punch

Table 3-3. IPSS Data Required to Model I/O Devices

Device Type	Data
Disk, Drum	number of packs per control unit
	number of cylinders per pack
	number of tracks per cylinder
	maximum track capacity
	maximum block size allowable for the device
	rotational speed
	data transfer rate
	cylinder access times
Tape	number of tape units per control unit
	tape recording density
	tape speed (reading/writing)
	inter-block gap size
	maximum block size recorded on the tape
	tape start-stop time
	forward erase length
	rewind rate
Unit Record (Card reader, punch, operator's console,etc.)	maximum block size
	transmission mode
	transmission rate

block size, and the number of records processed. An example is given in Figure 3-7. This example shows two files, one an unblocked card-image file of 554 records which is identified through the comment as SIDPERS file COOAAC. The other file, CICAAC, contains 987 506-byte records. Note that the block size specified in the AF Table is the unit of I/O for the application program and need not represent the secondary storage block size.

The Application Processing Table (AP Table)

The AP Table mimics the I/O processing done by an application program. Each table entry consists of two records, first a processing specification record, followed immediately by a processing definition record.

The specification record identifies the type of processing, (D for any delay due to the operator, I for input, P for CPU activity, and O for output). The "D" definition record quantifies the delay; the "I" and "O" definition records specify: the file, a concurrency index, random or sequential processing, and percentage of file processed; and the "P" definition record specifies the type of activity engaged in by the application program, such as EDIT, SORT, or REPORT.

Figure 3-8 depicts a typical example of a job step (for example, SIDPERS). First, there is a delay of 10 to 15 seconds due to operator responses to console messages, or tape mounts. Then all of file 01, and 50% of file 10 are read concurrently, file 01 sequentially and file 10 randomly; one of the files is edited, and the output is sent to file 04. Finally, 5% of file 10 is rewritten after all other processing is complete. Note that the order of application record processing is

File id	LRecl	Block Size	# Records	Comments
01	80	80	554	Card Image Input - C00AAC
10	506	506	987	Edit Table File - C1CAAC

Figure 3-7. An Example of the Application File Table

```
*
*  A Typical Jobstep in SIDPERS
*
   D
      1000.     15000.    1.0      TAPE MOUNT
*
   I                                C00AAC
        01  X  S  100.
*
   I                                C1CAAC
        10  X  R   50.
*
   P                                CPU PROCESSING
         X      EDIT
*
   O                                B1AAAC
       .04  X  S  100.
*
   O                                C1CAAC
        10  Y  R   10.
```

Figure 3-8. An Example of the Application Processing Table

determined by the concurrency indicator (the "X" and "Y" in Figure 3-8). The "X" indicates that files 01, 10 and 04 are processed concurrently, (i.e., read one record of file 01, one of file 10, write one to file 04, then repeat until all three files are exhausted). The "Y" of Figure 3-8 indicates that file 10 is written after the complete processing of files 01, 10, and 04. (Any alphanumeric characters, except blank, may be used as a concurrency indicator). As also shown in this figure, comments may appear on the right hand side of any data card, and on any "comment" card (which is designated by an asterick in the first column).

Description of Database Characteristics

The term database is used here to mean the data files managed by the hardware system's data files. Those files required by an application must be characterized by the Modeler and the results placed in the System File Table.

In this table, each record gives secondary storage information about a single file. Each file is assigned a volume type (disk, tape, or console) and a volume number. The logical record length, blocksize, and file size (number of logical records) are recorded. Disk file information includes the extent type (index (I), primary (P), or overflow (O)) the percentage of records on the primary extent (%PE), the number of secondary extents (#SE). If the file placement is known, it is given in terms of low and high cylinder and track addresses. (If unknown for disk files (U), the file is placed randomly on the volume during IAPS simulation). Finally a comment field is provided as an aid to the modeler. Provision is made to define VTOC files (V),

sort work files, and messages to and from an operator's console. For detailed formatting information, see Appendix F.

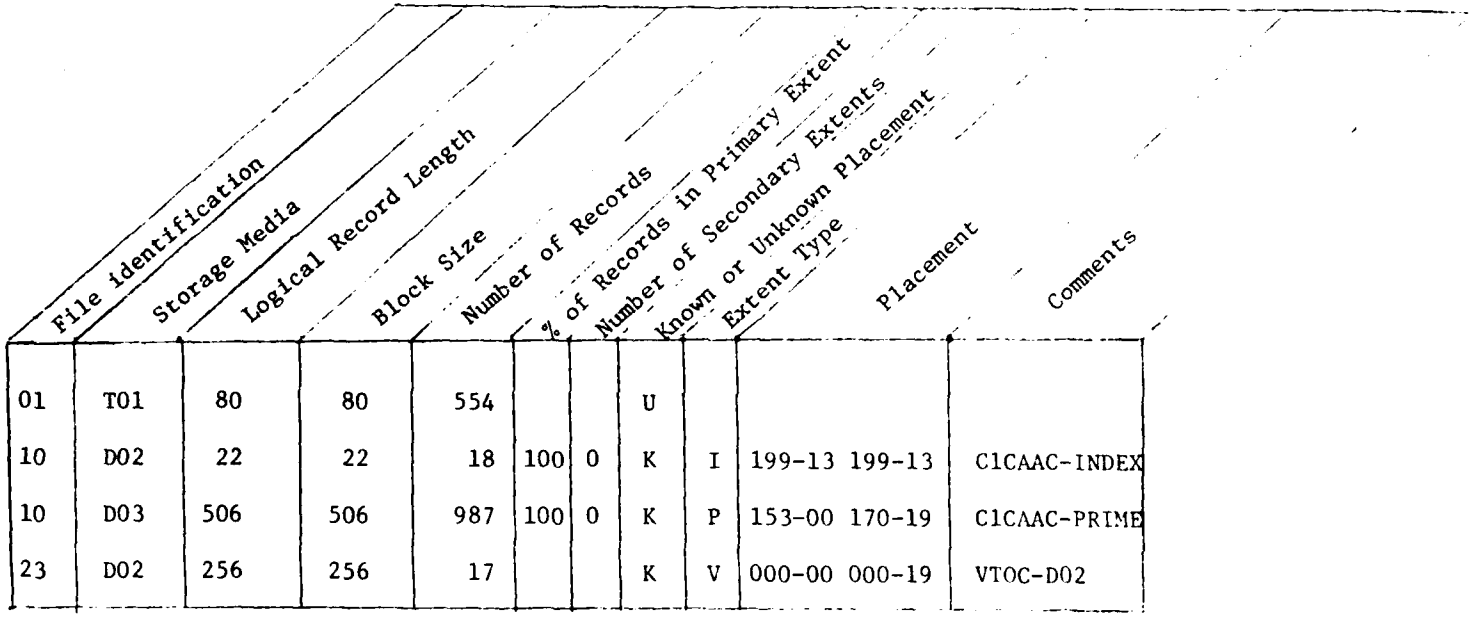
The examples in 3-9 are typical. System file 01 has 554 unblocked records with an LRECL of 80. It resides on tape unit T01 with an LRECL and blocksize of 80, and an "unknown" placement (U), which for tape files means that the file begins at the beginning of the reel. Note that the modeler has used comments to identify the file as COOAAC -card image input. User file 10 (the third line of Table 3-9), is also unblocked with an LRECL of 506; it has 987 records in its prime extent (P), which resides on disk unit D03, with allocated space from cylinder 153 track 0, to cylinder 170, track 19. This file is an ISAM file and thus has an index extent which resides on disk unit D02, giving among other things, its known placement.

3.4 THE IAPS SIMULATOR VIEW

The IPSS Simulator function requires a person or persons who are knowledgeable programmers and analysts of the IPSS language and execution facilities. The basic simulator role is to augment the IPSS model structure we have provided in order to be responsive to changing Modeler requirements.

A Simulator overview of the IAPS methodology is given in Figure 3-10. This figure shows the interaction among three components of IPSS: The Exogenous Event Stream Component. The IPSS Define Model Component is represented by the Equates between the two latter components.

We have completed the simulator function for the present IAPS methodology. A large class of computer systems can be simulated and



File identification	Storage Media	Logical Record Length	Block Size	Number of Records	% of Records in Primary Extent	Number of Secondary Extents	Known or Unknown Placement	Extent Type	Placement	Comments
01	T01	80	80	554			U			
10	D02	22	22	18	100	0	K	I	199-13 199-13	C1CAAC-INDEX
10	D03	506	506	987	100	0	K	P	153-00 170-19	C1CAAC-PRIME
23	D02	256	256	17			K	V	000-00 000-19	VTOC-D02

Figure 3-9. An Example of the System File Table

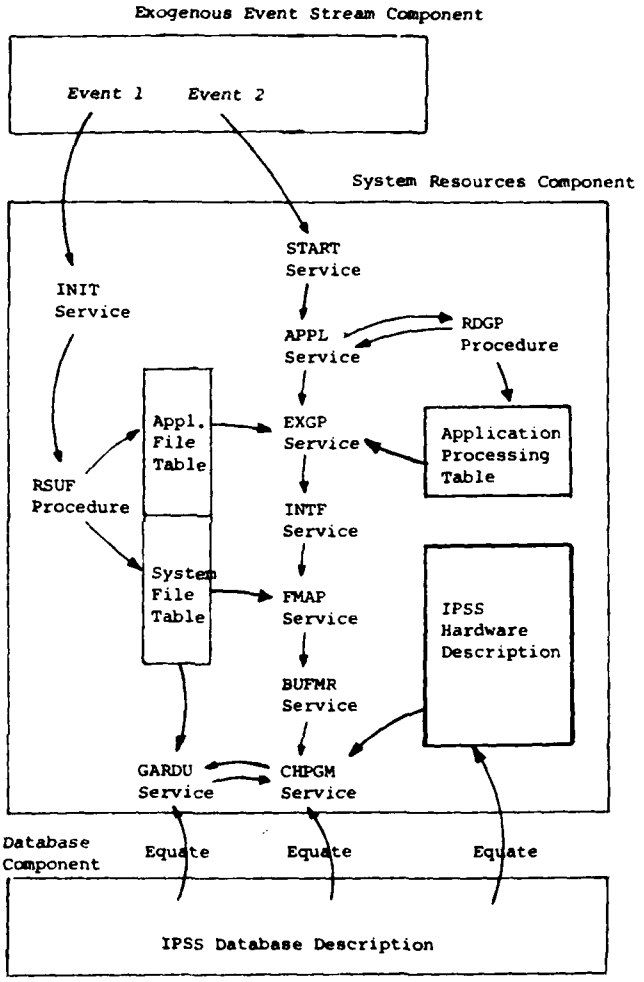


Figure 3-10. Overview of the Service Hierarchy in the IPSS Model - The Simulator's View of the IAPS Methodology

evaluated without any further Simulator activity. The Simulator is required if a system outside the scope of the present IAPS is to be modeled. Examples of such systems are: Database management systems, teleprocessing, distributed systems and operating system task management.

3.5 THE IPSS ANALYST VIEW OF COMPUTER SYSTEMS

The IPSS Analyst is a specialist in the IPSS Modeling and Execution facilities (see Appendix A). The IPSS Analyst view of the simulation process is represented in Figure 3-11. This role requires a knowledge of the details of the IPSS translation process, the simulation nucleus, and facility definition tables. The need for the IPSS Analyst will be further reduced over time as IPSS evolves into a more fully developed product. We required the type of knowledge represented by the IPSS Analyst role only a few times during the course of our project. Examples of what we required (and easily ascertained through source listings) were IPSS statement options, random number generation algorithms, and facility table value offsets.

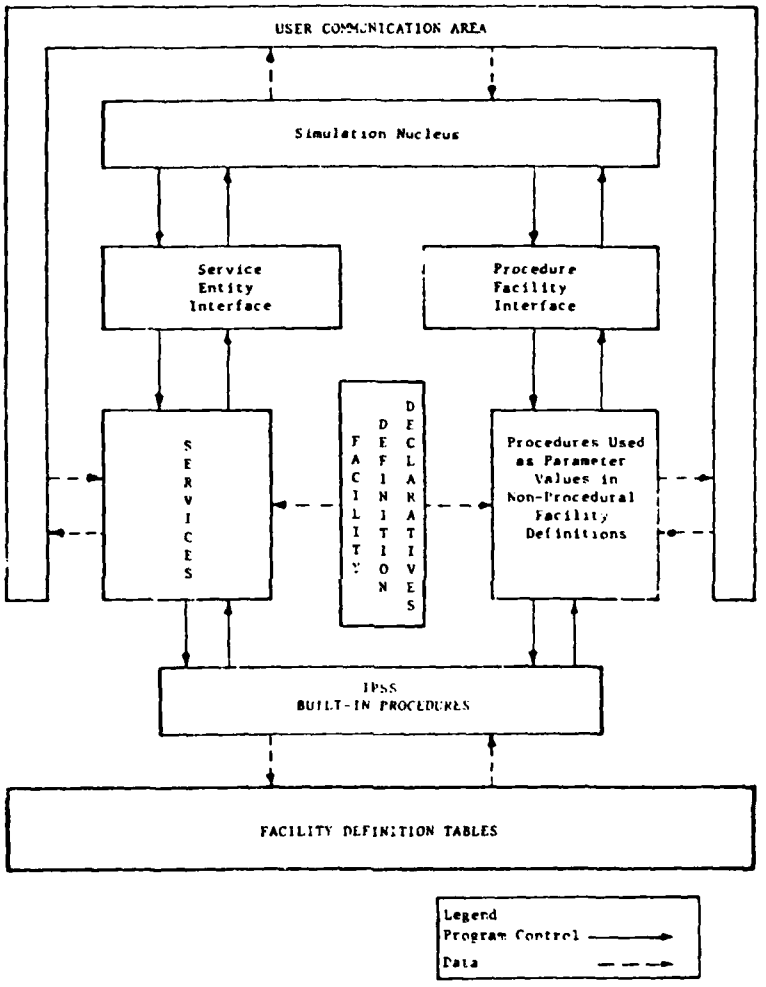


Figure 3-11. IPSS Program Control and Data Interfaces - The IPSS Analyst View of IPSS (DEL78a)

4. MODELING SIDPERS USING IAPS

4.1 SELECTION OF A SIDPERS SUBSET

SIDPERS is a standard, automated integrated personnel system designed to provide personnel information systems support at division, installation, brigade, battalion and unit levels (SID76). SIDPERS performs four major functions in support of Active Army personnel and organizations:

1. Strength accounting,
2. Organization and personnel recordkeeping,
3. Information exchange with other automated systems, and
4. Command and staff reporting.

A SIDPERS activity is designed to support a data base of computer files for up to 50,000 personnel and 1,000 organizations.

SIDPERS software consists of five DOS-E jobs:

- o AACR01 - Labels and Assignments
- o AACR02 - SIDPERS Basic Cycle
- o AACR03 - SIRCUS
- o AACR04 - SIDPERS Back-up Cycle
- o AACR05 - SIDPERS Recovery Cycle

The focus of our project was on the SIDPERS Basic Cycle, Job AACR02. This job consists of 19 job steps which proceed from editing functions, through file update, to reporting. Since the project duration and objectives did not permit the modeling and evaluation of all of SIDPERS, a subset of programs was selected with the assistance of USACSC Quality Assurance personnel (WHI79).

The editing programs, two of the master file update programs and one report preparation program were selected from all the programs in the SIDPERS basic cycle. Each editing program was a single job step and thus some validation data could be obtained. The selected update and report preparation programs were, however, single phases loaded and executed dynamically as part of a larger job step. While the modeling of the logic of these programs was not a problem, obtaining validation data for these phases was not possible. In addition, the modeling of the entire job step in which these phases were located would be almost as difficult, again, for lack of adequate validation data. Thus, the selected update and report preparation programs were not modeled.

The programs we modeled represent transaction classification, sorting, and validity editing; and incorporate concurrent direct and sequential access to disk files and sequential access to tape files. These programs are:

P1AAACA - transaction classification and scheduling,

P1BAACS - transaction sort,

P1CAAC - transaction validity editing, and

P1GAAACS - sort and update "queue" production.

For convenience and readability, these programs will be referred to as P1A, P1B, P1C, and P1G, respectively.

4.2 MODELING THE SIDPERS SUBSET

Once this subset of the SIDPERS programs was identified, we obtained current COBOL source listings, the DOS version of the SIDPERS

Operations and Scheduling Manual (SID79), and the SIDPERS Basic Cycle JCL listing. We analyzed these sources in order to obtain basic file data which is constant to any SIDPERS processing cycle. The type of data we obtained were file names, type of file (e.g., ISAM, sequential), use of file (input, ouput, both input and output), record processing mode (sequential or random). The details of our findings are summarized in Appendix C.

Next, we determined that the data we required for validating a model of SIDPERS was available from four sources, namely:

1. GRASP Accounting,
2. SYSLIST,
3. Operator Console Log, and
4. DITTO Utility.

The type of data provided by each of these sources is summarized in Table 4-1.

We visited the Ft. Stewart Division Data Center on August 2nd and 3rd, 1979, and observed the computer operation during SIDPERS Basic Cycle processing. We obtained computer listings for the data sources listed above. Table 4-2 presents a summary of the data we collected at Ft. Stewart for the first four job steps of the SIDPERS Basic Cycle on August 2, 1979, and indicates the source of the data items.

The following is a discussion of these data sources in more detail.

GRASP

GRASP provides a wealth of accounting data which is extremely useful in validating simulation models. For completeness, a list of the type of data available through GRASP is provided in Appendix G.

Table 4-1. Sources of System Data

1. GRASP Step Accounting

   CPU time
   Wait on operator time
   Job duration time
   Interference duration time
   I/O wait time
   I/O device usage time
   Start I/O counts
   Time waiting for and using the LTA
   SYSRES usage time
   Channel activity time

2. SYSLIST

   Gives job step start and stop time
   Number of records sorted
   Some file counts

3. Operator Console Log

   Number and length of console messages

4. DITTO Utility

   Record counts
   Type of records processed

Table 4-2. 2 August 1979 Ft. Stewart Record Processing for Programs P1A Through P1C

File Name	Media	Input/ Output	Concurrently Processed with	% File Processed	Number of Records	Source of Record Count Data
P1A						
C0OAAC	Tape	I		100	554	Card count
B1AAAC	Tape	I		100	2111	B1AAAC out + 31 Grade Changes
C0OAAC	Tape	I		100	661	Tape DITTO
A1AAAC	Tape	O		100	1249	SYSLIST sort count in P1B
E1AAAC	Tape	O		100	1171	Tape DITTO
B1AAAC	Tape	O		100	2080	Tape DITTO
C1CAAC	Disk	I/O			987	Program source and # transactions
XUTAAC (X=A,B,C, E,F,G,H)	Disk	I		0	-	Program source listing
P1B						
A1AAAC	Tape	I		100	1249	SYSLIST
B1AAAC	Tape	I		0	-	Monthly only
C1CAAC	Disk	I/O		.3	987	
A1BAAC	Tape	O		100	1249	SYSLIST
B1AAAC	Tape	O		0	-	(see above)
SORTWK1-5	Disk	I/O				Computed

Table 4-2 Continued.

File Name	Media	Input/ Output	Concurrently Processed with	% File Processed	Number of Records	Source of Record Count Data
P1C						
A1BAAC	Tape	I		100	1249	SYSLIST
C1CAAC	Disk	I/O		125	987	Source program and input transactions
A1CAAC	Tape	O		100	1249	SYSLIST
P1G						
A1CAAC	Tape	I		100	1249	SYSLIST
R1GAAC	Tape	I		100	50	
C1CAAC	Disk	I/O			987	
SORTWK1-6	Disk	I/O				Computed
X1GAAC (X=A,B,C, E,F,G,I,J, K,M,N,Q,R)	Disk	O		(A) 0 (B) 0 (C) 100 (E) 100 (F) 100 (G) 0 (I) 0 (J) 0 (K) 100 (M) 0 (N) 0 (Q) 0 (R) 0	- - 30 1210 7 - - - 2 - - - -	SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST SYSLIST

GRASP, however, was of limited use to us since GRASP step accounting was not available at the Ft. Steward Division Data Center. Table 4-3 summarizes the data we obtained from GRASP for the August 2, 1979, execution of the SIDPERS Basic Cycle. As shown in this table, GRASP job accounting statistics did not provide us with any useful data for programs P1A through P1G. Since the cycle we observed was initially cancelled in program P1G, we were able to use the GRASP CPU time of approximately twelve minutes as an estimate of the complete P1A through P1G CPU time.

SYSLIST

SYSLIST was of value in determining file record counts only when the job contained a sort. Program P1B and P1G sort entire files and the number of records sorted is reported on the SYSLIST.

Operator Console

The operator console log did not provide us with any data on the number of records processed. However, we observed that, because of the amount of time spent displaying and responding to messages, the operator's console was a more important element in the system from a performance perspective than we originally anticipated.

Tape DITTO

By far the most useful method of determining the number of records processed on a per file basis is the Tape Utility DITTO. This utility simply lists the entire file, allowing not only an accurate count but also insights into the types of data being processed. We obtained DITTO listings of the transaction input files and the stacker files.

Table 4-3. SIDPERS Basic Cycle, 2 August 1979, Ft. Stewart (Extracted from GRASP Accounting Reports)

Activity	Complete Cycle	P1A through P1G cancel	P1A through P1G complete
Elapsed time	4/33/27	38/54	17/50**
Non-MPS time	4/25/23*	38/53	N/A
CPU time	2/36/12	11/56	N/A
Pages spooled	165	11	N/A
Pages loaded	4443	363	N/A
Transient Area time			
Wait	10	0	N/A
Use	1/15/34	26/05	N/A
RES I/O	31622	1893	N/A
Operator console time	42/28	23/10	2/27***

*Does not include 07/11 restart time

** from SYSLIST

*** from Console log

5. MODELING TWO COMPUTER HARDWARE ARCHITECTURES IN IAPS

5.1 IPSS HARDWARE CHARACTERIZATION

This section discusses the modeling of the IBM 360 Model 30 computer (CS_3) and the Honeywell Series 60 Level 6 Model 47 minicomputer (DAS_3) systems. A computer architecture is typically represented in IPSS by characterizing the following:

1. the hardware devices, their capacities and processing characteristics,
2. the interconnections among the hardware devices, and
3. the operating system.

Hardware Devices, Capacities, and Processing Characteristics

The block diagram for the two modeled systems are shown in Figure 5-1 and 5-2. The connecting edges between primary and secondary storage represent the paths along which data is transferred. Note that in the 360/30, the dual channel tape controller allows either channel 1 or channel 2 to complete an I/O request. Thus, there are two paths to the tape units and one to the disk. We assume that channel 1 will be used to access a tape unit when channel 2 is busy.

The focal point of the IPSS modeling of these systems is on the secondary storage subsystem. We examined technical specifications provided by the respective vendors and extracted performance characteristics and capacities for both the direct access storage devices and the magnetic tape units. These characteristics are reported in Tables 5-1 and 5-2 respectively. This data was incorporated into

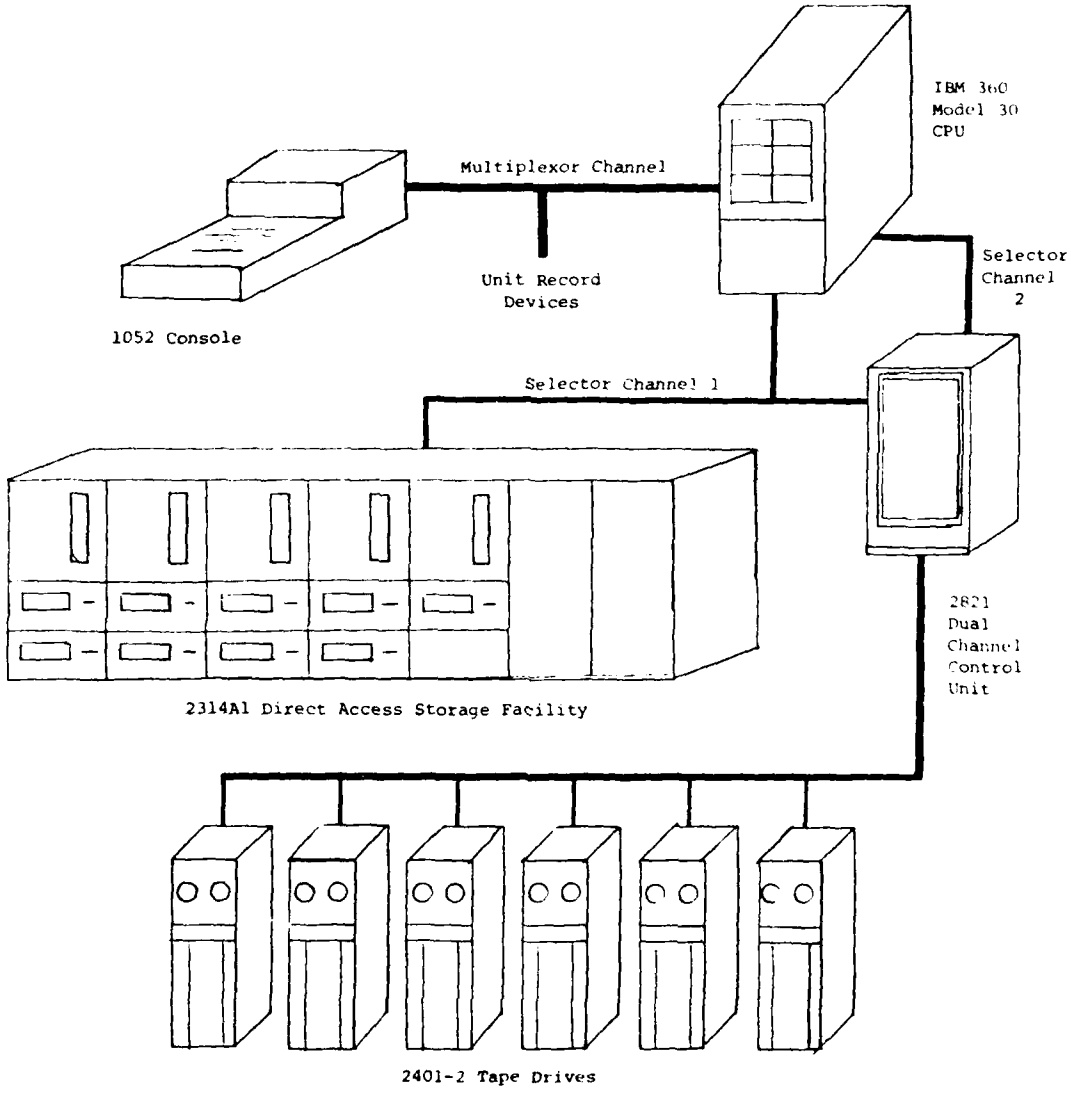


Figure 5-1. Typical CS_3 Hardware Configuration

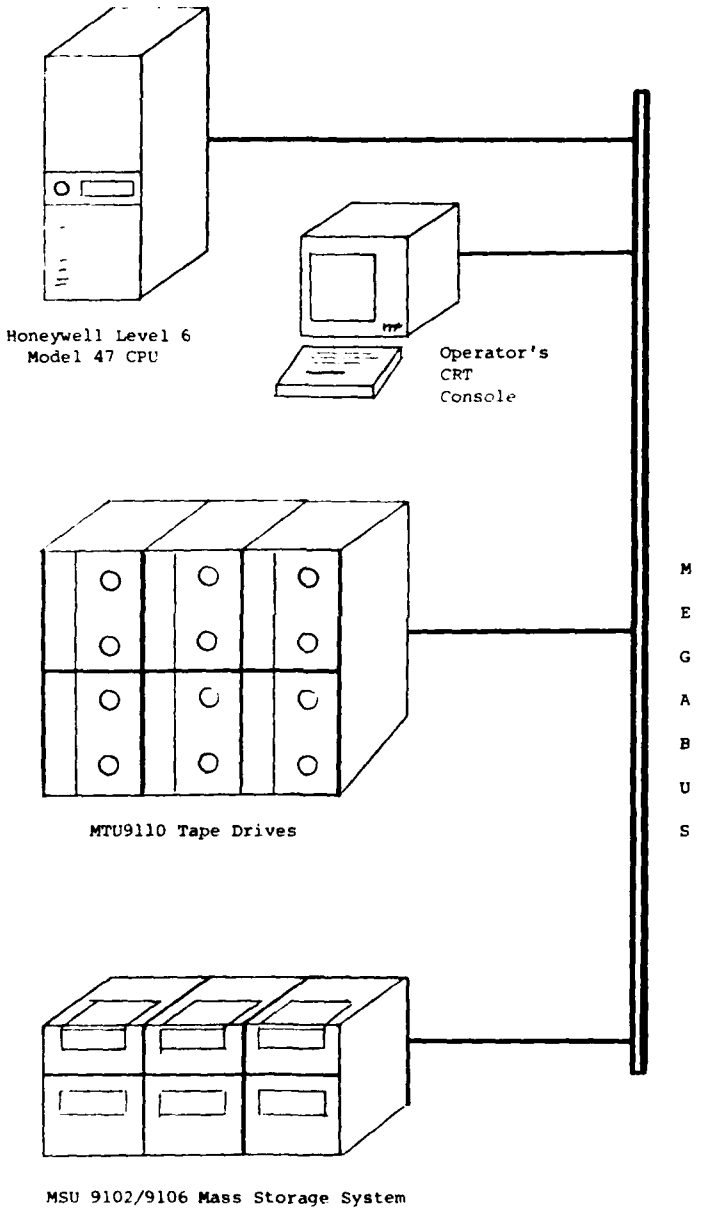


Figure 5-2. Honeywell Level 6 Architecture - DAS_3

Table 5-1. Direct Access Storage Devices

Physical Characteristics	Disk Units		
	IBM 2314A	IBM 3330	Honeywell MSU 9102/9106
Drives per unit	8	8	3
Bytes per unit	233.4M	800M	201M
Speed			
Average Seek	60 ms	30 ms	30 ms
Average Rotational Delay	12.5 ms	8.4 ms	8.33 ms
Data rate (kilobytes per second)	312	806	1,200
Capacity			
Cylinders per pack	200	404	823
Tracks per cylinder	20	19	5
Tracks per pack	4,000	7,676	4,115
Bytes per track	7,294	13,030	16,384*
Bytes per cylinder	145,880	247,570	81,920
Bytes per pack	29.18M	100M	67M

*Based on 64.256 byte-sectors per track

Table 5-2. Magnetic Tape Units*

Physical Characteristics	Tape Units		
	IBM 2400-1 Model 5	Honeywell MTU 9109	Honeywell MTU 9110
Drives	6	2	6
Tracks	9	9	9
Density	800/1600	800/1600	800/1600
Inter-block gap (inches)	.6	.6	.6
Block length	-	2048	-
Speed			
Read/write (inches per second)	75	45	75
Rewind rate (inches per second)	350	200	250
Data transfer rate (bytes per second)	120k	36k/72k	60k/120k
Start/stop time	13 ms.	8.33 ms	5 ms

*Where more than one characteristic is listed, the underlined number was included in the model(s).

IPSS models through IPSS hardware-facility statements. A sample of these statements is given in Appendix B.

Interconnections Among Hardware Devices in IPSS

In the IPSS System Resources component, device characteristics are associated with an access mechanism and volume. However, channels, control units, and the CPU are independent, unrelated facilities. These facilities are interrelated in IPSS models by a service which represents a channel program. This service, usually called CHPGM, is almost a standard part of every IPSS model and plays a central part in the IAPS methodology. Its function is to generate a physical (device) address and to seize, use and release all the facilities (e.g., CPU, channel controller, access mechanism, volume) in the path from the CPU to the secondary storage device in order to simulate a data transfer. The IPSS CHPGM service is listed in Appendix B.

Operating System Representation in IPSS

In IPSS, an operating system is represented by one or more services which simulate job scheduling, task management and resource allocation activities. We investigated but did not represent the operating system functions in either the IBM or the Honeywell model. The reason is that we did not have time to analyze these function, or model them, in sufficient detail to warrant their inclusion in the models.

However, our investigation revealed that the IBM 360 Model 30 supports a Disk Operating System (DOS) with the following major support packages:

- o GRASP accounting package,
- o DYNAM/T tape manager,
- o ADAS disk manager, and
- o SYNCSORT sort package.

We attempted to ascertain how these packages interact with DOS and under what conditions they request I/O. The next step would have been to represent processing, resource allocation, I/O characteristics, and dispatching of each of these packages (including DOS) in one or more IPSS Endogenous Services. Following this, we would include these services at the appropriate place in the IPSS model (i.e., at the INTF service), then verify and validate the resulting model.

5.2 ARCHITECTURAL VARIATIONS

For convenience in referencing the hardware systems that we modeled, we designated the model of the IBM 360 Model 30 as Model A1, and will refer to the model of the Honeywell Level 6 minicomputer as model B1. In addition, we considered three variations of model A1 and one variation of model B1 in order to demonstrate the capabilities of IPSS and the IAPS methodology.

As shown in Table 5-3, the variations on model A1 are the replacement of the 2314 disk unit with a 3330 disk unit (A2), the replacement of the 14 character per second operator console with a 960 character per second console (A3), and both of the above replacements (A4). These experiments were designed based on observations of the current 360 Model 30.

Table 5-3. Hardware Differences

Model Designation	Summary of Architecture Differences
A1	Standard 360 Model 30 o 14 characters per second operator console o 2314 direct access storage facility
A2	360 Model 30 o 14 characters per second operator console o 3330 direct access storage facility
A3	360 Model 30 o 960 characters per second operator console o 2314 direct access storage facility
A4	360 Model 30 o 960 characters per second operator console o 3330 direct access storage facility

Table 5-3 Continued.

Model Designation	Summary of Architecture Differences
B1	Standard Honeywell Level 6 Model 47 o 6 MSU9106 disk drives o 6 MTU9110 tape drives
B2	Honeywell Level 6 Model 47 o 3 MSU9106 disk drives o 2 MTU9109 tape drives

*All 360 Model 30 architectures had six 2400-1 Model 5 tape drives. All Honeywell Level 6 Model 47 architectures had a 960 character per second operator console.

The variation on model B1 was the deletion of three disk drives and the replacement of the 6 MTU 9110 tape units with 2 MTU 9109 tape units.

Table 5-4 shows the performance characteristics of these architectural variations relative to model A1 (the standard 360 Model 30). Model B1 is clearly superior to A1 in every way except tape concurrency (each has six tape drives), and all the variations show at least one area of superiority.

Table 5-4. Performance Characteristics of Alternate Hardware Architectures Relative to Model A1 (Standard 360 Model 30)

Performance Characteristic	Model Designation					
	A1	A2	A3	A4	B1	B2
Capacity						
Disk (bytes)	1	3.4	1	3.4	1.7	.9
Tape (available for concurrent use)	1	1	1	1	1	.3
Speed						
Disk* (I/O per unit time)	1	2	1	2	2	2
Tape** (I/O per unit time)	1	1	1	1	2.5	1.4
CPU (Instructions executed per unit time)	1	1	1	1	7.3	7.3
Operator Console (Characters per unit time)	1	1	68.6	68.6	68.6	68.6

*Based on average seek and average rotational delay and time to transfer one 100 byte record

**Based on time to transfer one 100 byte record and start/stop time

6. OVERVIEW OF IAPS MODEL STRUCTURE AND EXECUTION

An IAPS model consists of a collection of IPSS service facilities whose invocation sequence is hierarchial. Figure 6-1 depicts the relationships among the services and indicates their generic function. As shown in the Figure, the arrival of an application job on a computer is represented by the START service. Several different applications could be started simultaneously and, if so, would compete for systems resources (such as the operating system, main memory, data channels). In the models we synthesized, only one application was started, namely SIDPERS. The START service invokes the application processing service APPL and waits for its completion. The APPL service determines the processing required for an execution group (DOS job step), invokes the EXGP service to perform this processing and waits for its completion. The EXGP service represents the processing performed by a user-determined unit of work. This service is driven by the values provided by the RDGP procedure. Its main functions are to schedule I/O activities to data files, and to represent CPU processing. I/O is represented by an invocation of the INTF service and a wait for response. The INTF service is essentially a null routine which is a system link to future processing activities (such as DBMS or the operating system). Currently, INTF invokes the FMAP service which represents the mapping of the application files to specific system files which are located on secondary storage. A single application I/O request to FMAP will generate one or more requests for system records. FMAP issues a request for a system

record by invoking the BUFMR service and then waits for a response. BUFMR represents the buffer management function. If a system record is in one of the buffers, an immediate response to FMAP is generated, otherwise the CHPGM service is invoked. CHPGM represents channel program processing; it uses the IPSS data base structure and hardware facilities to generate a hardware address, computes the read/write time, and advances the simulator clock accordingly. Table 6-1 relates the IPSS services identified in Figure 6-1 to the corresponding SIDPERS processing activities.

Model Synthesis

One of the advantages of IPSS in general, and IAPS in particular, is the ease of synthesis of experiments. Figure 6-2 outlines our basic approach to producing models with different hardware architectures. We functionally divided the IPSS models and placed the parts into members of a partitioned data set (member names are in parenthesis in the Figure). We then chose members from each of the following categories to form a complete model:

1. Application processing
2. Architecture
3. Define Model
4. Loading

The Application Processing group is the nucleus of our IAPS methodology. It contains all the IPSS services and facilities of the System Resources Component except for hardware facilities and the channel program service. Also included in this group are the IPSS Exogenous Event Stream component and the Data Base Structure Component.

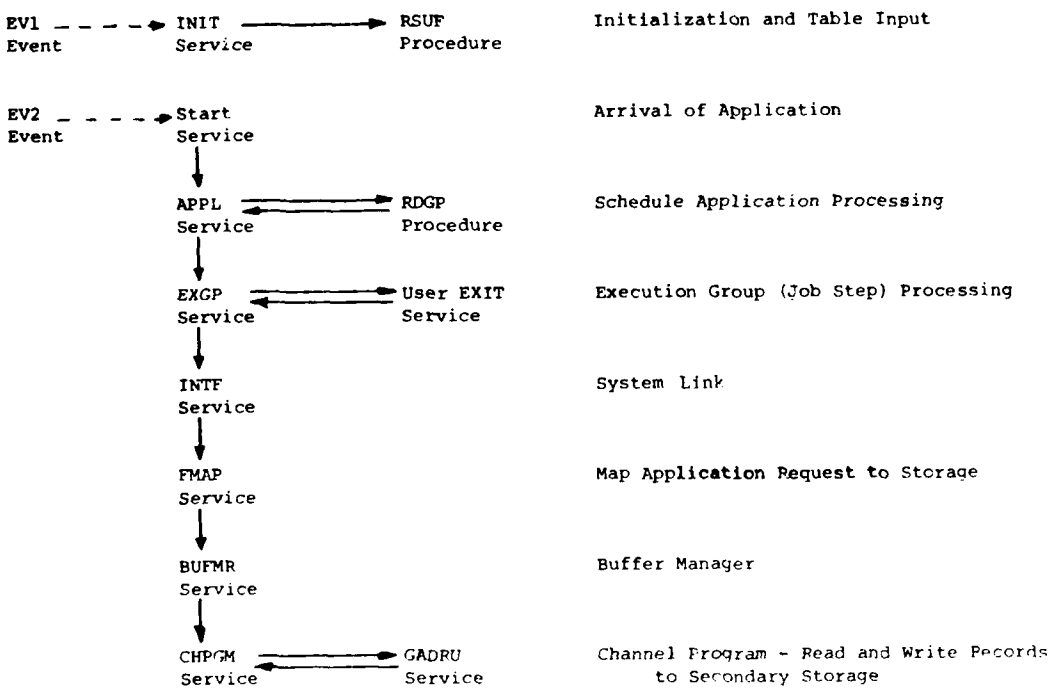


Figure 6-1. Simulator Overview of the IAPS Service Hierarchy

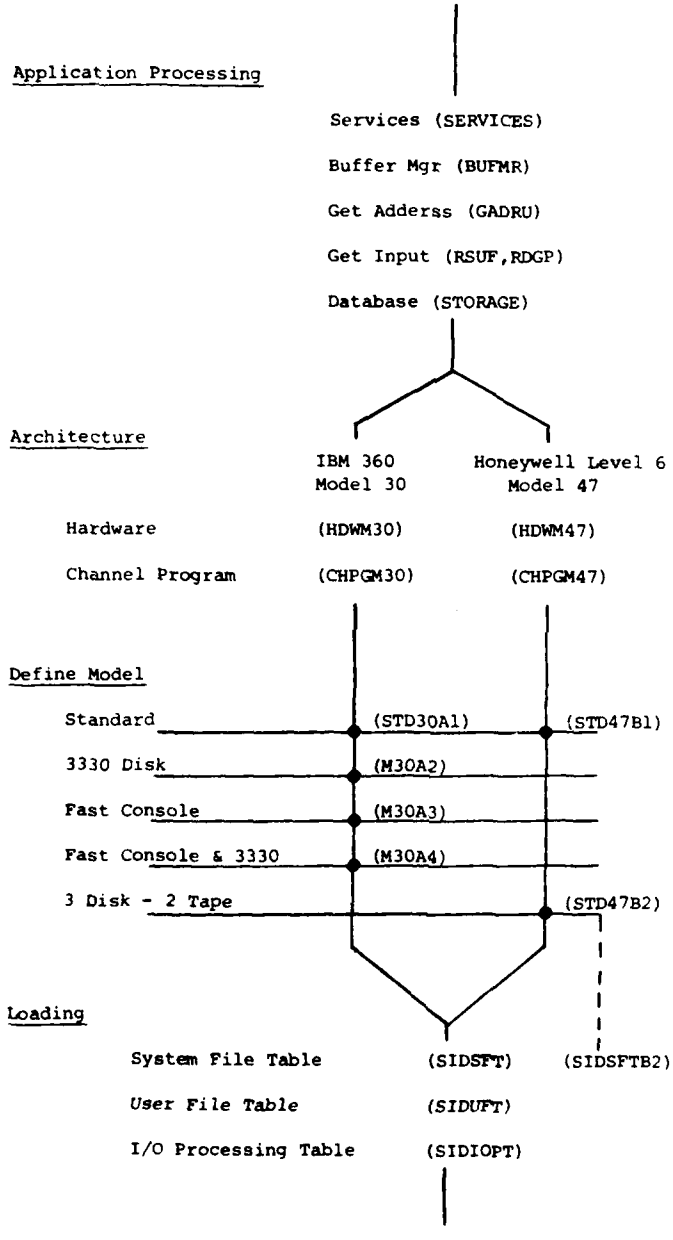


Figure 6-2. Summary of Models Synthesized for Evaluation of Alternate Hardware Configurations

Table 6-1. Index to Modeled SIDPERS Processes

Level	SIDPERS Process	IPSS Endogenous Exogenous Service Name
1	(IPSS initialization)	INIT
2	Arrival of SIDPERS job, job scheduling	START
3	SIDPERS processing	APPL
4	Job Step Execution	EXGP
5	Processing link for OS, DBMS, etc. (future use)	INTF
6	SIDPERS logical record to DOS physical record mapping	FMAP
7	Buffer management	BUFMR
8	Channel program, retrieval of records from secondary storage	CHPGM

The Architecture group contains services and hardware specific to the two generic classes of hardware systems being modeled, namely the IBM 360/30 and the Honeywell Level 6. One set (i.e., hardware specification and channel program) was selected for each execution of the model.

The Define Model group is the IPSS Model component. Each member of this group specifies a hardware alternative. Using these members, we were easily able to replace disk and operator console units in the model and to ascertain the effects on the overall performance of the system. The members of this group were easy to generate and use. Clearly this type of experimentation is one of the primary benefits of modeling in IPSS and using the IAPS methodology.

The last group of members which were selected were from the loading group which represents the external (i.e., SIDPERS) loading on the computer system. These can be easily modified to represent different application processing characteristics.

At the completion of this research project our program library contains members which represent (a) at least eight variations on the basic hardware of the IBM 360/30 (CS_3) system; (b) two variations of the Honeywell Series 60 Level 6 Model 47 (DAS_3) system; (c) a general model of computer software, including submodels of application programs, a buffer manager, and a channel program; (d) tables of data which describe in detail the files used by SIDPERS (Section 4.3); (e) a table which describes the sequence of I/O and processing performed by the first four job steps of SIDPERS, which table is processed by the first submodel mentioned previously in (c); (f) a command procedure in simple question and answer form which allows a user to put together and execute

a complete model from the library members described in a-e, and thus easily to compare design alternatives (Figure 3-4). For a listing of library member names and contents, see Appendix E.

These models were executed on an Amdahl 470 V/6-II with OS/MVS. Each model contained approximately 2800 lines of code (IPSS, Fortran, and comments). Each model required approximately 400KB of main storage and four minutes of CPU time (compilation plus execution). Modeling experiments were facilitated through the creation of load modules which were repeatedly executed. This reduced the simulation run time by approximately one minute.

7. ANALYSIS OF RESULTS OF THE IAPS SIMULATIONS

The ultimate purpose of an IAPS simulation is to present the decision-maker with data which will prove useful in the overall evaluation and comparison of alternative designs. The decision-maker will take many things into account which are not addressed by any simulation, such as the availability of appropriate compilers and the projected cost of maintenance. A valid simulation, however, can provide information which can be obtained from no other source except the much more expensive alternative of running the actual workload on the actual computer system. Examples of such information include answers to the following questions:

1. What is the projected run-time of SIDPERS on the DAS_3 system, and how does it compare to the existing system?
2. Which of the hardware resources is over-utilized, and thus potentially a bottleneck as system workload increases?
3. How will the system respond to an increase in workload over time?
4. How will the system respond if one or more tape, or disk, units become dysfunctional?

For the simulation user to have confidence in the results produced by any simulation, he needs a systematic approach to the validation and analysis of the output statistics of the simulation. It is our purpose in this chapter to outline such an approach in the context of our application of the IAPS methodology to the hardware

comparison of the CS_3 and DAS_3 configurations when run against the same SIDPERS workload. However, we were unable to carry out this approach in its entirety due to lack of data and lack of time.

In the following sections we discuss model verification and model validation, an analysis of the results of simulating the CS_3 and DAS_3 systems in six configurations, and the results of some additional simulations.

7.1 MODEL VERFICATION AND VALIDATION

Model verification is the act of testing the logic of the model to determine that it behaves as the simulator intended. In short, it is "debugging" the computer program. During the verification process, the model may be driven by real or imaginary data, but is usually driven by simplified data so that the modeler can follow the logic of the model in detail by hand calculations. We verified the IAPS model components by using a detailed trace which printed the occurrence of each event in chronological order and the value of any variable whenever it changed. Further discussion of verification techniques can be found in standard simulation texts such as those by Shannon [SHA75], or Fishman [FIS73].

Verification is to be distinguished from validation, which is the act of comparing the model to the existing system. As a part of our IAPS simulations, we compared the IPSS output statistics from the model of the standard IBM 360/30 to data collected at the

Ft. Stewart DDC on 2 August 1979 during an actual run of the SIDPERS application system. Results are presented in Table 7-1. All times are in minutes and seconds.

The first three rows of Table 7-1 give the actual data collected at the Ft. Stewart DDC and used for validation purposes. The first row gives the elapsed time for each job step (P1A, P1B, P1C, and P1C) and the total elapsed time for all four job steps, the source of this data being the SYSLST. Since we did not model the operating system, we adjusted the elapsed times of row one by an estimated time which represented the operating system's I/O to the SYSRES pack. The amount of SYSRES I/O was again known from the SYSLST. The adjusted elapsed times, row two of Table 7-1, thus provide the primary data for validation purposes. Operator console times, row three of Table 7-1, provide a secondary source of validation data. These times were computed by actually counting the number and lengths of messages on the console log from the 2 August SIDPERS run, and by using our observation of console speed, namely 11 characters per second and approximately 1/2 second for carriage return. (IBM rates their 1052 console at 14 characters per second, but our observations and timings indicated otherwise.)

Other types of system data useful for validation purposes include CPU busy and idle times, other resource utilization, and queueing information. Due to the lack of job-step accounting we were unable to obtain any validation data other than that in Table 7-1. It is also recommended that validation data be collected from more than

Table 7-1. Validation Results

	SIDPERS Job Step				
	P1A	P1B	P1C	P1G	Total
2 Aug '79					
Elapsed time (minutes: second)	4:59	2:33	9:23	3:49	20:44
Elapsed time less RES I/O	4:17	2:13	8:04	3:16	17:50
Operator Console* (computed)	1:40	:45	:30	:41	3:36
IPSS Model					
Elapsed time	4:12	1:59	8:0	3:8	17:19
% difference	-2%	-10%	-0%	-3%	-3%
Operator Console	1:35	:54	:25	:41	3:35
% difference	-5%	+20%	-17%	+0%	-.5%

*at 11 characters/second and 1/2 second carriage return

one run of SIDPERS, and if such data is used for model calibration purposes, that a second independent set of data be collected for validation purposes. Due to lack of time and the difficulty of obtaining such data, we were unable to obtain more than one set of data. Our identification of data sources (Table 4-1) should ease data collection in future studies.

Table 7-1 also gives IPSS output statistics of elapsed time and console times for the model of the IBM 360/30, plus percent difference between the validation data and the model data. As can be seen, overall elapsed time differed by only 3% and console time by less than 1%. Based on the limited data available to us, we accepted our model as valid.

7.2 ANALYSIS OF SIMULATION RESULTS

The main statistic of interest in our simulation study was job (and job-step) elapsed time. These results are presented in Tables 7-2 and 7-3.

Table 7-2 presents the simulation elapsed time per job step and the total elapsed time for all four job steps for four variations on the IBM 360/30 system and two variations on the Honeywell system. All of the decreases in elapsed time except for B2 over B1 are to be expected, judging by the hardware characteristics and comparisons presented in Tables 5-1 through 5-4. The decrease in elapsed time of B2 over B1 (about 1 minute, 4 seconds) is due to a different placement of certain files. In its first four job-steps, SIDPERS has 8 tape files, and models A1, A2, A3, A4 and B1 model these files

Table 7-2. Simulation Elapsed Time per Job Step (minutes:seconds)

Experiment	SIDPERS Job Step				
	P1A	P1B	P1C	P1G	Total
A1 - Standard CS_3	4:12	1:59	8:0	3:8	17:19
A2 - fast disk	4:12	1:53	7:30	2:45	16:20
A3 - fast console	2:38	1:05	7:36	2:27	13:46
A4 - both	2:38	1:0	7:05	2:04	12:47
B1 - DAS_3	1:24	0:42	2:22	1:08	5:36
B2 - DAS_3(2 tape, 3 disk)	1:02	0:22	2:21	0:47	4:32

Table 7-3. System Comparison

Base System	Alternate System		SIDPERS Run-time on Alternate System * (hours:minutes)
	System**	Alternate System time/Base System time	
A1 (Standard CS_3)	B1	.32	2:34
	B2	.26	2:05
	A2	.94	7:31
	A3	.80	6:24
	A4	.74	5:55
A4 (CS_3 with fast console and 3330 disk)	B1	.44	3:31
	B2	.35	2:48

*Assumes a base system run time of 8 hours

**B1 - Honeywell Level 6 Model 47 (DAS_3)
B2 - DAS_3 with 2 tape, 3 disk units
A2 - CS_3 upgraded by 3330 disk
A3 - CS_3 upgraded by fast console
A4 - CS_3 with both 3330 and fast console

as tape files. For model B2, however, the last six of the tape files are placed on disk. (The remaining two tape files are the SIDPERS input transactions.) As can be seen by examining Tables 5-1 and 5-2, the average time to transfer a block of data is faster for disk than for tape.

Two things should be kept in mind when examining Table 7-2. First, we did not model the availability of storage space. No claim is made that any configuration (especially B2) will be adequate for the storage of SIDPERS files. Second, we assumed that all tapes are premounted and that the operator takes no more than 10 seconds per job-step to respond to console messages. These two assumptions are consistent with our observations at Ft. Stewart. However, premounting of tapes may become more difficult on a faster system (e.g., B1) or impossible on a smaller system (e.g., B2 with only 2 tape units).

Keeping these limitations in mind, plus the restriction of our model to the first four jobsteps of SIDPERS, we can make a few tentative conclusions based upon Table 7-2. We see that the best upgrade of the CS_3 system, namely A4, improved performance in terms of elapsed time by approximately 25%. On the other hand, either of the two DAS_3 configurations improved performance by approximately 70% or more.

Table 7-3 presents a comparison of system A1 to its alternates, and a projection of SIDPERS run time. For example, the first four jobsteps of SIDPERS ran on system B1 in 32% of the time required on A1.

If the first four job-steps were representative of all of SIDPERS, then we would predict that an 8 hour SIDPERS run on A1, the IBM 360/30, would take 2 hours and 34 minutes on B1, the Honeywell Level 6 Model 47 (with 6 disk and 6 tape units). We emphasize that the right-hand column of Table 7-3 should not be taken as a firm prediction, but is merely for illustrative purposes. Such a prediction could only be made after modeling all or at least a substantial portion of SIDPERS. Table 7-3 also contains comparisons of the "best" upgraded version of A1, namely A4, to the two Honeywell configurations, B1 and B2.

The results in Tables 7-2 and 7-3 are illustrative of the type of results and comparisons that can be made when evaluating computer systems. Similar comparisons could be made of other quantities of interest, such as resource utilization, queueing times for resources under contention, and response time in an interactive environment.

7.3 ADDITIONAL EXPERIMENTS PERFORMED

To demonstrate the ease of model building after a library of model components is in place, we made a number of additional simulation runs.

The purpose of the first set of runs was to investigate the variability of the estimate of elapsed time due to the random elements in the model. In all experiments, random access was modeled by picking the next record to be read (or written) in a random fashion, by making use of the GGU3 random number generator, a routine in the IMSL mathematical and statistical subroutine package which is documented in [LEA73]. Another source of randomness was the random placement of files on disk when their

placement was unknown. The result of these and other uses of the random number generator is to make the estimate of elapsed time a random variable.

For experiment A1, three independent runs were made using three independent sources of random numbers. (Run 1 in each case is the run presented in Table 7-2.) The results are presented in the first 3 rows of Table 7-4. As can be seen, elapsed times for runs 1 and 3 were identical (when rounded to the nearest second), and the elapsed time for run 2 differed by only 1 second in job-step P1C. This lack of variability of the estimate of elapsed time increases our confidence in its precision. Table 7-4 also presents the results for 2 independent runs each of experiments B1 and B2. Similar conclusions can be drawn from these results.

The purpose of the second set of runs (A5, A6, A7, and A8) was to demonstrate the ease of building models from an existing library. All of the eight additional runs in Table 7-4 were made by recombining existing elements of the library, and took less than one hour to submit from an interactive terminal using the technique illustrated in Figure 3-4. All four of these models represented upgrades of the CS_3 system (A1). In experiment A5, the 2314 disk units were replaced by 3340 disks. In A6, the memory cycle time was reduced by 50% to measure the effect of doubling CPU speed. In A7, six of the eight tape files in the first four jobsteps of SIDPERS were placed on disk, so that model A7 faced the same loading as did model B2. Finally, model A8 was identical to model A4 but its loading was that of A7 and B2.

Table 7-4. Simulation Elapsed Time per Job Step

Experiment	Run	SIDPERS Job Step (Elapsed time in minutes:seconds)				
		P1A	P1B	P1C	P1G	Total
A1	1	4:12	1:59	8:0	3:08	17:19
	2	4:12	1:59	7:59	3:08	17:18
	3	4:12	1:59	8:0	3:08	17:19
B1	1	1:24	0:42	2:22	1:08	5:36
	2	1:24	0:42	2:22	1:08	5:36
B2	1	1:02	0:22	2:21	0:47	4:32
	2	1:09	0:22	2:24	0:47	4:42
A5*		4:12	1:51	7:17	2:39	15:59
A6		3:25	1:46	5:04	2:30	12:45
A7		3:35	1:41	7:31	2:47	15:34
A8		1:57	0:40	6:35	1:41	10:53

*A5 - A1 with 3340 disk

A6 - A1 with memory cycle time reduced by 50%

A7 - A1 with 2 tape files (loading identical to that for B2)

A8 - A4 with 2 tape files

Again we emphasize that the systems modeled whose results are exhibited in Table 7-4 were chosen only to illustrate the ease of model building when using the IAPS methodology and the types of results which a valid simulation can give a decision-maker.

8. SUMMARY OF PROJECT ACTIVITIES

Three people were assigned to this project for the methodology and model building tasks. In addition, IPSS maintenance support was provided by a half-time undergraduate student. Work began on approximately 14 June 1979 and continued through 14 September 1979. The IPSS models of the IBM and Honeywell computer systems were developed, verified and validated as of 21 August 1979.

Excluding the half-time student, a total of 189 man-days were authorized for this project, of which approximately 180 were used. Table 8-1 provides a breakdown by major category. Twelve days were spent in developing the methodology and 24 in determining what validation data was available at which computer installations. This is considered to be a one-time cost. The User activities took 18 man-days, exclusive of documentation. The Modeler activities consumed 38 man-days, 25 of which were in examining SIDPERS. Fifty-six days were spent developing the IPSS code for the IAPS methodology, and 5 days were spent at the IPSS Analyst level of detail. Documentation consumed 21 days and project start-up used 5 days.

Table 8-2 compares the current research project with estimated man-day costs for several different continuing projects of similar [illegible] scope. The first column gives the actual man-days for the current project, and is taken from Table 8-1. Our first projection [illegible] would essentially be a continuation of the current project. It [illegible] to run additional simulations with already existing members of the library, or wanted to consider additional minor variations on the [illegible] IAPS system, we project

Table 8-1. Breakdown of Project Activities

Methodology	
o Design of IAPS methodology	12
o Determine availability of validation data	24
	36
User	
o Determine architectures and variations to be modeled	2
o Execute IPSS models from libraries	1
o Validation	5
o Analysis and Evaluation	10
	18
Modeler	
o Characterize IBM 360 Model 30 in IPSS	4
o Characterize Honeywell Level 6 in IPSS	9
o Develop SIDPERS processing characteristics (site visit, study COBOL programs and console logs and SYSLIST and tape DITTO, encode data)	25
	38
Simulator	
o Develop IPSS routines to input application processing tables	10
o Develop IPSS routines to process application processing tables	21
o Code and verify the overall structure of the IPSS model	25
	56

Table 8-1 Continued.

IPSS Analyst	
o Study the internal logic of IPSS on a special-case basis	5
Miscellaneous	
o Project start-up time	6
o Documentation	21
	27

Table 8-2. Man-Power Analysis and Projection

Activity	Current Research Project	Projections in Man-Days			
		Further CS_3, DAS_3 SIDPERS Evaluation	Evaluation With All of SIDPERS	Different Hardware and Software	Operating System, CS_3 DAS_3
Methodology	36	0	0	0	30
User	18	10	10	15	15
Modeler	38	0	20	40	45
Simulator	56	0	0	15	50
IPSS Analyst	5	0	0	0	5
Start-Up and Documentation	27	10	10	10	25
Total Man-Days	180	20	40	80	170

a man-day cost of 10 days for running the simulations and analyzing the results, plus 10 days for start-up and documentation.

The second project we consider, of slightly greater scope, consists of comparing the CS_3 and DAS_3 systems with all of SIDPERS as the workload. Modeler activities would consist of a projected 20 days to examine the relevant COBOL application programs and to translate the sequence of I/O processing into the IAPS Application Processing Table, to collect the necessary data and to encode it into the System File Table and Application File Table; and finally to add these new members to the library. User activities would then consists of a projected 10 days for making runs and analyzing the results, plus 10 days for documentation.

The third project involves the development of the capability to model hardware other than the CS_3 and DAS_3 systems, and to model additional application software such as STANFINS. The addition of new hardware capabilities would require a projected 20 days of Modeler activities and 15 days of Simulator activity. Specific tasks to be performed would include characterization of the new hardware, data collection, coding of the data into IPSS statements, the writing of a channel program, and the addition of these new members to the library. The modeling of additional software would be a project of approximately the same scope as the second project. In summary, this third project, with a total of 80 projected man-days, would be of a scope similar to the current project, but would require 100 fewer man-days of effort becuase of our previous development of the IAPS methodology and the pre-existing library of model components

which represent application (I/O) processing and thus can be used with any hardware configuration or any new application software.

The fourth proposed project consists of extending the currently existing models to include operating system components. The current models do not include a representation of the operating system. New methodology would have to be developed, taking a projected 30 man-days and involving persons highly familiar with the operating systems for the IBM 360/30 (namely, DOS-E) and with that for the Honeywell system. Modeler, Simulator, and IPSS Analyst activities would require a projected 100 man-days. Specific tasks would include extensive consultation with operating system experts and data collection, plus the development of IPSS code to represent the operating system. User activities to run the model, validate it, and evaluate the output would take a projected 15 days, plus a projected 25 days to document the new members of the library and the simulations performed. At least 90 of the total projected 170 man-days would be one-time costs, after which the operating system model components would be available in the model library for future use.

Provided that an extensive library of model components were built up and maintained, future projects of the scope discussed here would tend to take less time than projected. The building and maintaining of a large and extensive model library of various hardware and software components is the key to providing timely answers to those questions which can be addressed by simulation.

9. SUMMARY AND CONCLUSIONS

SUMMARY

This project was an intensive, short-term research and development effort focused on the simulation of computer systems for the U.S. Army Computer Systems Command. Specifically, we addressed the problem of providing a model development tool which would be responsive to meeting Command simulation objectives. This required a methodology for model development, use, and analysis which would be easy-to-use, widely applicable to many types of computer systems, amenable to change, and time-efficient.

We designed, developed, implemented, and tested a methodology to meet these objectives. Our methodology, called IAPS (IPSS Application Processing System), structures the modeling process into hierarchical levels which identify specific tasks in the modeling cycle. These levels are named for the person or persons who are responsible for the activities defined within a level. A User is responsible for the overall evaluation effort. He produces an evaluation of a specific computer performance problem through (a) interactive model building in an easy question and answer format (which results in the submission of a model for computer execution) and (b) analysis of the results. The procedure for building a model uses building-block components from a model library. The role of the User presupposes that a Modeler has provided the appropriate building-blocks and has made them available for the User in the model

library. The Modeler characterizes computer hardware, the software for application processing systems, and the data files. These latter two elements are characterized in a language that we designed and implemented as part of this project. In the role of Simulator, we also wrote the program which translates these characterizations into performance statistics. The Information Processing System Simulator (IPSS) language served as our base. IPSS provides specially designed built-in hardware and software language statements which greatly facilitated our programming task. We completed the Simulator's task for a large class of application processing systems. Further effort, however, is required for modeling advanced features such as data base management systems, operating system functions, and teleprocessing.

This methodology was applied to an existing U.S. Army software system (SIDPERS) run on several IBM and Honeywell computer configurations. As Modelers, we visited an operational computer installation and collected data on a SIDPERS daily cycle. We also determined performance specifications on the IBM Model 30 computer and the Honeywell Level 6 minicomputer. This software and hardware data was encoded into IAPS source statements. Then, as Users, we built models using our interactive approach and conducted a set of experiments to analyze the performance of several architectural variations, all executing with the standard SIDPERS workload. We verified and validated our model of SIDPERS and its execution environment (an IBM 360 Model 30 computer). We then projected execution times for SIDPERS on a Honeywell Level 6 minicomputer. Our results reflect the faster CPU and peripherals of the Level 6 minicomputer.

We varied the type and speed of the peripherals on both systems to demonstrate the responsiveness capabilities inherent in our IAPS methodology. Our primary measure in evaluating these alternative configurations was total elapsed time to run the SIDPERS job. We also obtained queuing and resource utilization statistics since these are automatically generated by IPSS.

CONCLUSIONS

The objective of this project was to produce a model building methodology for simulating U.S. Army computer hardware/software systems. The project definition required a demonstration of our methodology by building models of an existing as well as a future U.S. Army computer system.

We designed our methodology based on our perception of current Army simulation needs. We implemented the methodology using the Information Processing System Simulator (IPSS), and we tested it using a subset of the programs in the SIDPERS basic cycle. Two major conclusions can be drawn from our efforts. One relates to the use of IPSS in modeling U.S. Army computer systems, and the other relates to the IAPS methodology for expressing application processing software and files. We conclude that IPSS is an appropriate tool for simulating the type of computer systems found within the U.S. Army. These systems are typified by a single processor, supporting either uniprogramming or multiprogramming, with I/O oriented COBOL file processing applications. IPSS incorporates special language features for characterizing computer hardware and files which make it especially

suited for the performance modeling and evaluation of these systems. The IPSS "service" concept helps the Simulator to produce a structured solution to complex model design problems.

The IPSS methodology and language proved to be relatively easy to learn and use. Two of the three researchers involved in this project had no prior IPSS modeling experience. With a few tutorials and IPSS models as a guide, they became productive IPSS modelers in a short period of time. The services of an IPSS expert, however, were required throughout the project.

Although IPSS is a prototype system, no IPSS source code had to be changed to generate the results produced in this report. We did identify enhancements, however, and these are detailed in the next section.

Our second major conclusion is that the IAPS methodology, and our implementation of its concepts, is an appropriate and useful method for characterizing U.S. Army computer hardware/software systems. Using IAPS, we were able to represent many types of file processing quickly and easily. In addition, the representation of computer hardware and the use of this hardware during file processing is one of the recognized advantages of IPSS.

We specifically designed IAPS with the objective of flexibility, generality, ease of use, and responsiveness. We tested and revised the methodology and the implementation during the project to more completely satisfy these objectives. We demonstrated flexibility and generality by modeling two different types of computer systems and several hardware variations. We incorporated ease of use by a library building

block approach to model synthesis and an interactive dialogue. We verified the responsiveness of the IAPS methodology by modeling some of the variations on short notice.

RECOMMENDATIONS

Our recommendations focus on three areas. First we present our recommendations for (1) further development of the IAPS methodology; (2) use of the methodology by the Computer System Command for further modeling of computer systems; (3) enhancement of the IPSS system itself.

IAPS Recommendations

Our experience as a User of the IAPS methodology suggests that it could be extended to allow a more sophisticated dialogue during model synthesis. We recommend the generation of library members from parameters input by the user. For example, the User could enter a small number of parameters for a sort operation, and the IAPS could generate the appropriate library member for this particular sort file processing. This enhancement would speed the modeling process by increasing the flexibility and generality of the library members. In addition, the interactive model synthesis could have an option such as "tutorial mode" to guide the novice model builder in great detail through every step of building a model from the model library. Such an addition to IAPS would greatly increase its ease of use and make model building a self-taught procedure.

The IAPS methodology could also be extended to include the simulation of data base management systems, operating system processing

and networks of computers. These would increase the scope of the methodology as well as the accuracy of the results obtained.

In addition, the IAPS methodology could be enhanced to allow the Modeler to represent computer hardware as he now represents computer files. This would allow greater flexibility in accommodating variations of hardware characterizations during experimentation.

Recommendations on the Use of IAPS

We recommend a continuation of the modeling effort which began with this project. In particular, we recommend modeling more of the SIDPERS basic cycle in order to further test the methodology and to verify our projections. This study should produce insights into selecting representative subsets of large systems for modeling and analysis.

We recommend the establishment of model libraries incorporating common computer architectures and software systems. This will enable the Computer Systems Command to respond quickly to future simulation needs.

We also recommend an IAPS simulation study be undertaken which involves a hardware modification. This would involve simulation and measurement of the system before and after the modification. This type of study would provide insights into the computer modeling process as well as a validation of the IAPS approach. The result would be increased confidence in the results of this type of simulation study.

IPSS Recommendations

With minor exceptions, IPSS proved to be a useful and appropriate tool for our modeling purposes. Many of our recommendations for the improvement of IPSS are already recognized and are in the process of being remedied. In particular, we make the following recommendations:

1. IPSS contains few implemented features for modeling CPU activity. Since circumstances did not permit us to model the CPU in any detail, this problem did not have a major impact upon our project, but may indeed affect any future modeling projects.
2. We were forced to rely on existing models and statement parsers to determine which options of the IPSS source code have been implemented. We were provided with a preliminary copy of a document that would remedy this situation, but its numerous errors rendered it useless. The corrected version of this document should be published, however, in the near future.
3. IPSS provides only 10 seeds to a random number generator and better random number generators are known to exist. This limits the number of independent experiments one can run to 10. We included a better random number generator and programmed a routine to accept a seed as input to the model and write out the last seed on model termination. These changes should be incorporated as a standard part of the IPSS package.
4. IPSS does not allow the modeler to save the load module and to execute the load module as a separate job (IPSS abnormally

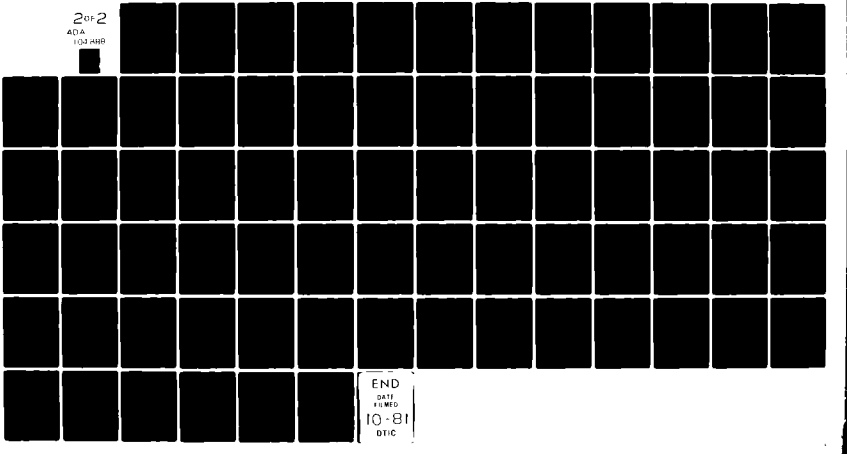
terminates when we tried this). As a consequence, every time an experiment is to be performed, the IPSS source language compilation and Fortran source language compilation process must occur. This consumed at least one minute CPU time for our models. We wrote special routines to bypass this problem. These routines should be incorporated as a standard part of the IPSS package.

5. We could not conveniently model concurrent activities since the IPSS automatic save/restore feature was not present in our copy of IPSS.
6. We could not declare data sets with a BLOCK reference unit due to an error in the Fortran built-in $CRDS routine. However, we were able to work around this error.
7. IPSS does not automatically collect statistics on UNIT RECORD or UNSPEC type devices. CREATE DATA SET and GET ADDRESS are two very useful IPSS built-in routines that only work on disk and tape devices. We modeled the operator's console as a Tape device to easily generate the utilization statistics we wanted.
8. A final area of possible enhancement of IPSS lies in the presentation and choice of statistical results. If desired by the user, quantities such as elapsed time should be converted from the simulation time unit (e.g. milliseconds) to hours, minutes, seconds. It also should be possible to have results tabulated and printed both cumulatively and over user specified intervals. We modeled SIDPERS at the

AD-A104 888 BATTELLE COLUMBUS LABS OH F/G 9/2
AN IPSS-BASED MODEL-BUILDING METHODOLOGY FOR RANKING AND EVALUA--ETC(U)
OCT 79 J D BROWNSMITH, J S CARSON

UNCLASSIFIED NL

2 OF 2
AD A
104 888



END
DATE FILMED
10-81
DTIC

LIST OF REFERENCES

ADL75 Adler, M., M. Wright, B. Conway and D. Carroll. 1975. SIDPERS Computer Simulation Model. Technical Documentary Report, U.S. Army Computer Systems Command. USACSC-AT-75-05.

BRO77 Brownsmith, J.D. and L.S. Chandler. 1977. Technical Report to the U.S. Army Computer Section Command.

DEL77 DeLutis, T.G., J.D. Brownsmith and J.S. Chandler. 1977. An IPSS Model of SIDPERS/IDMS. Technical Report to the U.S. Army Computer Systems Command.

DEL77 DeLutis, T.G. 1977. A Methodology for the Performance Evaluation of Information Processing Systems. Final report to National Science Foundation, OSIS GN 36622.

DEL78a DeLutis, T.G. 1978. Information Processing System Simulator (IPSS): Syntax and Semantics, Volumes 1 and 2, *working document to* NSF Grant 36622.

DEL78b DeLutis, T.G. 1978. A Simulation Language for Evaluating Information Processing Systems from a DBMS and User Perspective: Extensions to the Information Processing System Simulator. Technical Report to the U.S. Army Research Office.

FIS73 Fishman, G.S. 1973. Concepts and Methods in Discrete Event Digital Simulation. Wiley.

LEA73 Learmonth, G.P. and Lewis, P.A.W. 1973. Naval Postgraduate School Random Number Generator Package LLRANDOM, NPS55LW73061A, Naval Postgraduate School, Monterey, California.

MIH76a Mihram, G.A. 1976. Four Questions Regarding the Credibility of Simulation. Modeling and Simulation 7(2): 1225-1230.

MIH76b Mihram, G.A. 1976. Four Further Questions Regarding the Credibility of Simulation. Modeling and Simulation 7(2): 1231-1234.

ROS78 Rose, L. 1978. Computer Systems/Database Simulation. Technical Report to the U.S. Army Computer Systems Command - AIRMICS.

ROS79 Rose, L. 1979. IPSS: A Language and Methodology for Information Processing System Simulation. Simulation - Management Workshop, Atlanta, Georgia, p. 142-174.

SCH77 Schaaff, H. 1977. Description of the VIABLE Transactions for the DIMUI Model. USACSC.

SID76 SIDPERS User Manual. 1976. SIDPERS Guide for Commander and Staff. DA Pam 600-8-8.

SID79 Standard Installation/Division Personnel System (SIDPERS) Operations and Scheduling Manual DOS version. 1979. United States Army Computer Systems Command. USACSC Manual 18-1-B-AAc. Volume III (DOS).

SHA75 Shannon, R.E. 1975. Systems Simulation, the Art and the Science. Prentice-Hall.

SWE76 Swenson, H. et al. 1976. Simulation of the Six Standard Army Multicommand Management Information Systems (update). Part I final report SIDPERS. SAI COMSYSTEMS, McLean, Va. Technical Report to the U.S. Army Computer Systems Command.

VAN69 VanHorn, R. 1969. Validation. In the Design of Computer Simulation Experiments (Ed.: T.H. Naylor). Duke University Press. Durham, North Carolina.

WHI79 Whitaker, R.M. 1979. Meeting Among Representatives of AIRMICS, Contract Researchers and QAD. Memorandum for Record. 2 July 1979.

WHI79 White, P. 2 July 1979. Memorandum to Director, Quality Assurance, AIRMICS.

APPENDIX A

AN OVERVIEW OF THE INFORMATION PROCESSING SYSTEM SIMULATOR (IPSS)

This Appendix highlights the IPSS methodology for characterizing salient features of information processing systems, the IPSS simulator, and the IPSS execution facility. This Appendix was extracted from previous reports prepared by Dr. L. L. Rose, Assistant Professor, The Ohio State University (ROS78, ROS79).

A.1 THE IPSS METHODOLOGY

IPSS provides a methodology which, although specific to computer systems, is general in nature, and quite flexible. It affords the user a viewpoint from which he can construct a simulation model of any computer system at any level of detail desired. This methodology separates the characterization of a complex information processing system into separate, inter-connected components. It gives structure and direction to the user, who has the difficult task of defining just what it is he wishes to model.

Figure A-1 illustrates the role of the IPSS methodology in the design and simulation of an information processing system. We observe that IPSS provides the modeler a top-down approach to the definition of models. At the top of this figure we denote the loose connection of user system knowledge into a set of data and concepts that describe the Information System. This definition may be concise and complete, showing complete knowledge of the system and processes to be modeled; it may be very vague in all respects; it may be specific with regard to certain aspects and non-specific with regard to other aspects of the information system. It is the role of the IPSS methodology to enable the modeler, who possesses varying degrees of information about the information system, to construct a model at appropriate levels of detail to satisfy his modeling needs.

The IPSS methodological view is to characterize any information

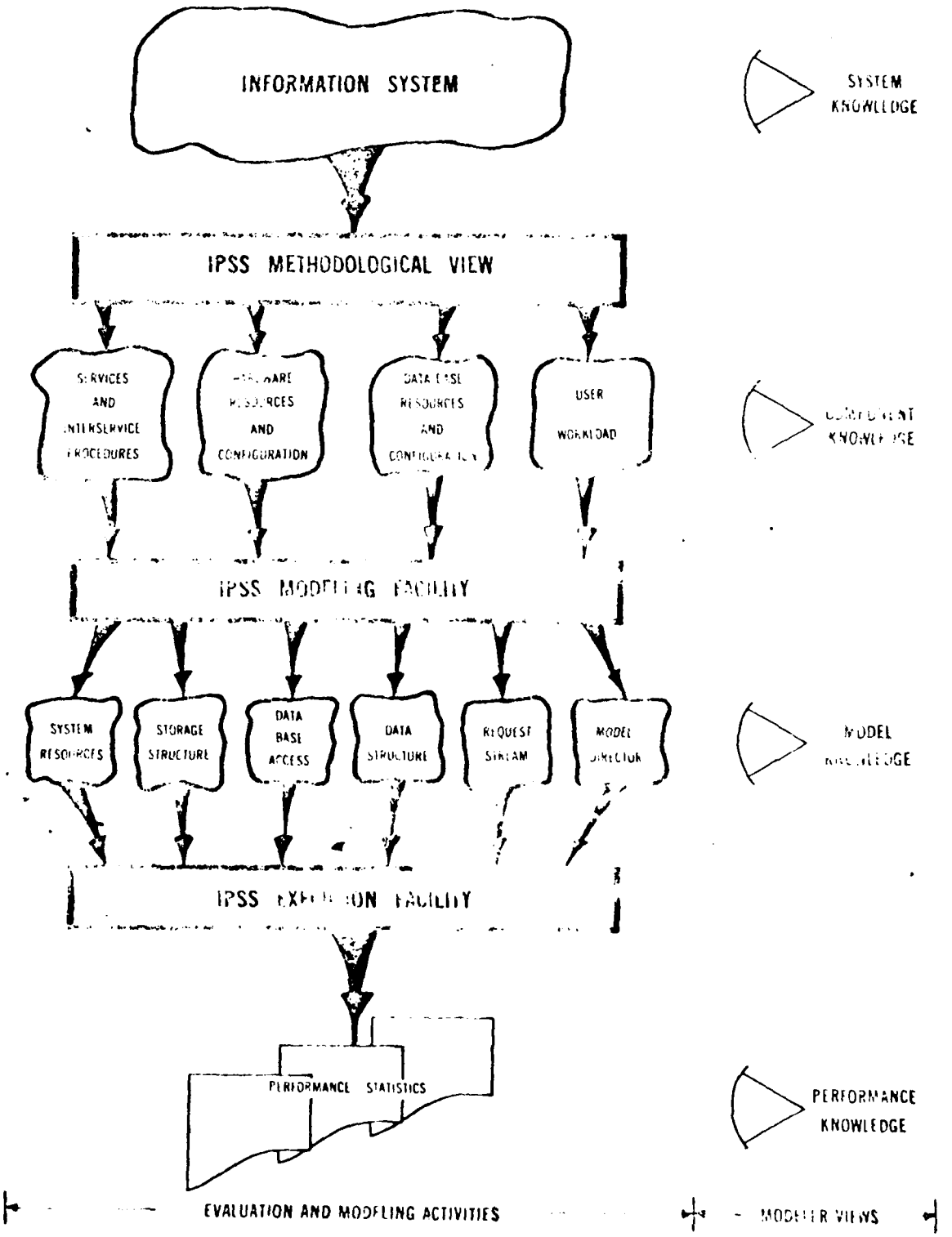


Figure A-1. The IPSS Methodology

processing system as a collection of four discrete but interfacing components. As illustrated in Figure A-1 these components are: 1) services and inter-service procedures, 2) hardware resources and configurations, 3) data base resources and configuration, and 4) user workload. These four component definitions are sufficient to characterize any information processing system; in particular, computer-based information systems or manual systems can be described.

Services and Inter-Service Procedures

The identification and definition of services and inter-service procedures is an important IPSS contribution, and separates its methodology (and subsequent modeling activities) from other systems such as DIMUI and CASE. A service procedure defines a task - manual or automatic - associating all related actions and times to complete the task. In a computer-based system, this component corresponds to the definition of all system software facilities, to include user application programs, the operating system, and the data management system. Service definitions, of course, are constrained to the level of detail required by the modeler or to the level of knowledge of the modeler. This is true of all four component definitions, and forces the modeler to realize the level of detail appropriate, and to obtain additional information, if required, to properly define each component. Note that no computer programming is being performed at this time; we are structuring the model to be defined and isolating user information into the

appropriate sets of component knowledge. In any computer programming activity, too much emphasis cannot be placed on structuring the prototype, for correct and appropriate structure can be followed by easy implementation which, by design, should reflect the needs of the modeler.

Hardware Resources and Configuration

The hardware resources and configuration component directly reflects the hardware system to be modeled. This component defines the CPU, primary storage, tapes, discs, drums, printers, terminals, channel controllers, etc., and all hardware interconnections. Again, the level of detail required is that appropriate to the goals of the modeling activity.

Database Resources and Configuration

The database resources and configuration component defines the logical database of the system to be modeled, to include schemas, file characteristics, database access capabilities, and user data access and data manipulation facilities. This component can reflect a current system with normal non-integrated file management or a future system with fully integrated data management capabilities.

User Workload

Last, but certainly of great importance, is the user workload component. It is here that one characterizes the workload to be placed on the simulated system, to include workload description, timing of inputs, files referenced, etc. This completes the structuring of the user's knowledge of the information system and

can be defined functionally or statistically.

A global view of the resultant component is as follows: work (input) to the information processing systems emanates from the user workload and requires certain services. These services may require other services (inter-service procedures) to perform the work required. Whenever database accesses are required, the database resources and configuration component defines and simulates logical data flow while the hardware resources and configuration component simulates the resultant physical data flow. This is the user's view of the information flow process at the conceptual level, structured into components by the IPSS methodology.

A.2 THE IPSS MODELING FACILITY

Given the user's component knowledge as structured by the IPSS methodology, this is transformed by the modeler into model knowledge using the IPSS modeling facility. This portion of IPSS also provides structure and modularity to the model definition, but at a realizable level, as opposed to the conceptual level of component knowledge. The result of this transformation from component to model knowledge is an IPSS-defined simulation model that can be executed by the IPSS execution facility.

There are six model components which comprise the resultant defined model. Given the separation of user knowledge into the four conceptual components defined previously, it is a straightforward task, conceptually, to define the six IPSS model components

which describe system resource, storage structure, database access, data structure, request stream and model director. To actually implement these modules represents a non-trivial, sophisticated effort that requires not only a good understanding of the system to be modeled, but also a complete understanding of how to effectively simulate all of the concepts and interactions of the process to be modeled.

IPSS provides a general simulation language and host environment to ease this task for the modeler. The Model Director is supplied for the user, and, in effect, directs the simulation defined by the other five model components. It handles the time clock, and the events queues, and all arrivals and departures from the system during model simulation. CASE and DIMUI effectively pre-define the entire simulation model (especially the system resources model component). This results in much less understanding about the model; it is the IPSS premise that a modeler cannot effectively use a simulation model that he does not understand.

As a result, IPSS offers a set of language constructs so that the user can, with relative ease, define all important aspects of the simulated activity. Using the IPSS statements, and any additional FORTRAN the user may desire, a FORTRAN model is output from the IPSS translator which can be executed to produce statistics. Additional FORTRAN statements are utilized by the modeler to either add statistics unavailable from IPSS or to model concepts not realized by the IPSS language constructs. In most cases, little additional FORTRAN is required as

IPSS provides a rich set of language constructs with associated statistical capabilities.

The top-down, modular approach provided by the IPSS enables the user to define, using IPSS/FORTRAN statements, five separate model components to characterize the system to be modeled. These are summarized below:

1. System Resources - Contains definitions for all information system resources (hardware and software) and all system tasks (application and operating system). This component forms the basic discrete event digital simulator for the information systems model under investigation. Included in the SYSTEM (system resources component) is the IPSS supplied clockwork mechanism to schedule and control simulated events and to determine when the simulation is to terminate. The clockwork logic is based on the next most immediate event philosophy for controlling discrete event digital simulations.

IPSS statements which ease the modeler's task of defining all of the system resources pertinent to the simulation desired include: Access Mechanism, Area, Buffer Pool, Central Processor, Control Unit, Data Channel, Data Set, Device, Endo Service, Exo Service, I/O Processor, Main Storage, Path, Procedure, Queue, Reference, Semaphore, Task, and Volume statements.

2. Storage Structure - Describes an information system's physical data base storage structure and its space management policies. The STORE (storage structure) component interfaces with the SYSTEM component in three ways. First, it references

SYSTEM to obtain Device and Volume facility definitions. Second, it supplies SYSTEM with Data Set facility definitions. Third, it translates secondary storage references specified as a displacement within a data set's logical address space into physical addresses within the secondary storage address space. Prior to a simulation, associations must be specified for the Data set, Organization Method, Device and Volume facilities. A STORE Organization Method facility can be associated with a multiple number of SYSTEM Data Set facilities. The opposite is true for the Device and Volume facilities. STORE Organization Method facilities are the templates from which the equated SYSTEM Data Set facilities derive their definitions during a simulation. The transfer of definitions between components is accomplished via the execution of the CREATE DATA SET Statement. The space management descriptions in STORE are used to calculate secondary storage addresses dynamically during a simulation based on facility definitions specified in each component and on the changes of these facilities during the course of the simulation.

IPSS statements provided to help the modeler define the Storage Structure Component include: Area, Segment, Organization Method, Extent, Record Type, Device, Procedure, Reference, and Volume.

3. Request Stream - Characterizes the information system's service request stream. It is responsible for the generation of all exogenous events for a model. Whereas SYSTEM contains facilities which characterize the processing requirements for each service offered by an information system, the request stream component (REQUEST)

defines the arrival of requrest for these services. IPSS converts these times into a composite arrival time stream.

The modeler thus defines exogeneous events, and IPSS eases this task by offering the Exogenous Event statement and the Procedure Statement should the modeler desire to define inter-arrival times functionally.

4. Data Base Access - Contains the definitions of all the resources required by the DBMS. These include the hardware resources of buffers and user work areas as well as application programs and DBMS software. All DBMS related entity-type facilities are defined within the component. The Data Base Access Component (ACCESS) is similar to the SYSTEM components in that it contains its own simulation clockwork mechanism similar in purpose to the one belonging to the REQUEST component.

IPSS statements particular to the Data Base Access Component include: DML Service, Realm, Schema, Record Origin, Semaphore, Task, and Queue.

5. Data Base Structure - Provides the modeler with a set of facilities which allows the definition of logical data structures and the characterization of relationships among them. This can be applied to a variety of DBMS architectures and application environments. The Data Base Structure component (STRUCTURE) permits the modeler to investigate the effects on system behavior caused by alternate set, record type, and access path definitions. The definitional facilities provided allow the modeler to investigate a wide spectrum of logical data structure organizations and allocation policies.

Within the Data Base Structure Component are IPSS statements to enable the modeler to define the following important database constructs:

Realm, Schema, Extent, Record Type, and Set.

A.3 THE IPSS EXECUTION FACILITY

The six IPSS model components discussed in the previous section (MODEL being pre-defined while SYSTEM, STORAGE, REQUEST, ACCESS, and STRUCTURE are user-defined with the aid of IPSS language constructs) comprise the input to the IPSS Execution Facility. It should be under tood, however, that this six-component model definition serves not only as necessary input to the IPSS Execution Facility. Of at least equal importance is the fact that the user has now created a documented, readable, understandable definition of the system to be modeled. The fact that this model is explicitly defined at user-determined levels of detail for each model component means that we have a hard copy description of exactly what the modeler wishes to simulate. No implicit assumptions (such as are contained in CASE and DIMUI) exist; hence user verification of the model can be accomplished much more effectively, and the entire modeling effort is at the level of detail desired by the modeler.

The IPSS execution facility carries out the simulation as defined by the six IPSS model components. This execution requires translation of IPSS statements into FORTRAN, link-editing of all required object modules, saving certain user-requested object/source modules in the IPSS library, and executing the resultant load module. Were the user required to define to the computer this multi-step job, a great deal of JCL (machine-dependent job control language) would be

necessary. In fact, both CASE and DIMUI require the user to create his own multi-step jobs, a non-trivial, machine-dependent task. The IPSS philosophy is to remove the tedium and complexity of JCL from the user; in fact, the user specifies no JCL whatsoever to execute an IPSS model. Thus IPSS must contain, within its own code, this JCL. We find this within the IPSS Nucleus, which is written in Assembler language. Hence we find that the IPSS is not completely portable, but only the Nucleus must be re-written to enable execution on another dissimilar machine.

A.4 THE IPSS STATISTICS

IPSS provides a modeler with a number of statistics concerning the behavior of modeler defined entities and IPSS supplied built-in information system services. Many output statistics are provided by IPSS automatically; others can be generated by the modeler's use of IPSS commands to start/end data collection on queues, facilities, services, etc. The IPSS-defined (automatic or modeler invoked) output statistics fall into eight general categories:

1. Operational Statistics,
2. Request Stream Statistics,
3. I/O Activity
4. Queueing Statistics,
5. Utilization Statistics,
6. Wait Statistics,
7. Service Statistics, and
8. Task/Activity Statistics.

Additionally, the modeler can employ the complete facilites of the FORTRAN language to develop his own statistics. Statistics are printed automatically at the conclusion of each model simulation unless explicitly inhibited.

APPENDIX B

IAPS SOURCE CODE

This Appendix contains examples of IPSS source code. Specifically, it contains a complete listing of the IPSS System Resources component for the IBM 360/30 (and all variants considered in this project). The System Resources component gives specifications and characteristics of all hardware components in the model. Following this are three examples of IPSS Services, which are used to represent software and application program I/O and CPU processing.

```
                                INPUT CARD IMAGE                                  INPUT    ALT INPUT  MEMBER
.....10.....20.....30.....40.....50.....60.....70.....80                          SEQ-NO   LEV SEQ-NO REF-NO

 DEFINE SYSTEM RESOURCES: NAME = SID360M30,                                  00000010      1
                          COMPILE = YES,                                     00000020      2
                          DISPOSITION = KEEP;                                00000030      3
                                                                             00000040      4
 PROCEDURE: NAME =  MENT, TYPE = SUBROUTINE;                                 00000050      5          1
C      MODEL OF - USACSC SIDPERS PROCESSING SYSTEM                           00000060      6          2
C      FOR - U.S. ARMY COMPUTER SYSTEMS COMMAND                              00000070      7          3
C      WRITTEN BY - JOSEPH D. BROWNSMITH, JOHN S. CARSON II, AND             00000080      8          4
C                   WILLIAM HOCHSTETTLER III                                 00000090      9          5
C      DATE - JULY-AUGUST 1979                                               00000100     10          6
C      HARDWARE - IBM 360 MODEL 30                                           00000110     11          7
C      NOTE - THIS MODEL IS WRITTEN IN THE IPSS LANGUAGE. IPSS               00000120     12          8
C             IS THE INFORMATION PROCESSING SYSTEM SIMULATOR                 00000130     13          9
C                                                                            00000140     14         10
C      THIS MODEL CONSISTS OF THE FOLLOWING COMPONENTS -                     00000150     15         11
C   REF    SERVICE/PROCEDURE     APPROX     INVOKES                          00000160     16         12
C    #           NAME            PAGE       CALLS        DESCRIPTION        00000170     17         13
C   ---  --------------------  ----       ------  ----------------------    00000180     18         14
C                                                                            00000190     19         15
C    1   SYSTEM RESOURCES        (01)         -      COMPONENT DEFINITION    00000200     20         16
C    2     COMMENT (PROC)        (01)         -        COMMENTS              00000210     21         17
C    3     INIT (EXS)            (07)       13,12      RD SYS FILE (EVI)     00000220     22         18
C    4     START (EXS)           (09)         5        START APPL PROCESSIN  00000230     23         19
C    5     APPL (ENS)            (11)       14,6,8     APPL PROCESSING       00000240     24         20
.....10.....20.....30.....40.....50.....60.....70.....80
```

Figure B-1. An Example of IAPS Source Code

```
                                INPUT CARD IMAGE                                                    INPUT    ALT INPUT  MEMBER
.......10........20........30........40........50........60........70........80                      SEQ-NO   LEV SEQ-NO REF-NO

C     6     INTF (ENS)          (23)        7       SYSTEM INTERFACE     00000250                   25                  21
C     7     FMAP (ENS)          (24)       11       APPL -> FMS MAPPING  00000260                   26                  22
C     8     USER (ENS)          (29)        -       USER PROC EXIT       00000270                   27                  23
C     9     BUFMR (ENS)         (30)        9       FMS BUFFER MGM       00000280                   28                  24
C    10     GADRU (ENS)         (38)        -       GET DATA SET ADDRESS 00000290                   29                  25
C    11     CHPGM (ENS)         (57)       10       FMS CHANNEL PGM      00000300                   30                  26
C    12     BUFST (ENS)         (63)        -       PRINT BUFFER STATS   00000310                   31                  27
C                                                                        00000320                   32                  28
C    13     RSUF (PROC)         (69)        -       READ APPL PROC TBLS  00000330                   33                  29
C    14     RDGP (PROC)         (84)       16       READ EXEC GROUP      00000340                   34                  30
C    15     IODC (PROC)         (96)        -       READ I/O DEFINITION  00000350                   35                  31
C    16     DDC  (PROC)        (102)        -       PROCESS DELAY DEF    00000360                   36                  32
C    17   STORAGE STRUCTURE    (105)        -     COMPONENT DEFINITION   00000370                   37                  33
C    18     DSAR (FAC)          (105)        -       DISK AREA FACILITY   00000380                   38                  34
C    19     DDSI  (FAC)         (107)        -       DISK DATA SET FAC    00000390                   39                  35
C    20     TAPAR (FAC)         (108)        -       TAPE AREA FACILITY   00000400                   40                  36
C    21     TDSI  (FAC)         (110)        -       TAPE DATA SET FAC    00000410                   41                  37
C    22   EXO EVENT STREAM      (113)        -     COMPONENT DEFINITION   00000420                   42                  38
C    23     EV1 (EXO EVENT)     (113)        -       GET SYS FILE INPUTS  00000430                   43                  39
C    24     EV2 (EXO EVENT)     (114)        -       START APPL PROCESSIN 00000440                   44                  40
C    25   MODEL                 (115)        -     COMPONENT DEFINITION   00000450                   45                  41
C                                                                        00000460                   46                  42
C                                                                        00000470                   47                  43
      RETURN                                                             00000480                   48                  44

.......10........20........30........40........50........60........70........80
```

INPUT CARD IMAGE .....10.....20.....30.....40.....50.....60.....70.....80		INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
END: PROCEDURE;	00000490	49		45
	00000500	50		
	00000510	51		
CENTRAL PROCESSOR: ID=CPU;	00000520	52		46
	00000530	53		
MAIN STORAGE: ID=MSTOR,	00000540	54		47
SIZE = 256000;	00000550	55		48
	00000560	56		
INPUT OUTPUT PROCESSOR: ID=(SELECT,2),	00000570	57		49
MAX TRANSFER RATE = 800000;	00000580	58		50
	00000590	59		
INPUT OUTPUT PROCESSOR: ID=MUX,	00000600	60		51
MAX TRANSFER RATE = 200000;	00000610	61		52
	00000620	62		
CONTROL UNIT: ID = CU2314,	00000630	63		53
MAX TRANSFER RATE = 312000;	00000640	64		54
	00000650	65		
ACCESS MECHANISM: ID = (DK2314,8),	00000660	66		55
VOLUME = VL2316(1,8),	00000670	67		56
DEVICE = DV2314;	00000680	68		57
	00000690	69		
VOLUME: ID = (VL2316,8),	00000700	70		58
DEVICE = DV2314;	00000710	71		59
	00000720	72		

.....10.....20.....30.....40.....50.....60.....70.....80

```
                                INPUT CARD IMAGE
........10........20........30........40........50........60........70........80
DEVICE: ID = DV2314,                                                 00000730
        TYPE = DASD,                                                 00000740
        BLOCK SIZE = (VARIABLE, MAX(TRACK CAPACITY)),                00000750
        CYLINDERS = 200,                                             00000760
        TRACKS PER CYLINDER = 20,                                    00000770
        TRACK CAPACITY = (7294, CHARACTERS),                         00000780
        ROTATIONAL SPEED = 2400,                                     00000790
        TRANSFER RATE = 312000,                                      00000800
        SPACE OVERHEAD = 0,                                          00000810
        CYLINDER ACCESS = PROC(CYLACC);                              00000820
                                                                     00000830
ACCESS MECHANISM: ID = (DK3330,8),                                   00000840
                  VOLUME = VL3336(1,8),                              00000850
                  DEVICE = DV3330;                                   00000860
                                                                     00000870
VOLUME: ID = (VL3336,8),                                             00000880
        DEVICE = DV3330;                                             00000890
                                                                     00000900
DEVICE: ID = DV3330,                                                 00000910
        TYPE = DASD,                                                 00000920
        BLOCK SIZE = (VARIABLE, MAX(TRACK CAPACITY)),                00000930
        CYLINDERS = 404,                                             00000940
        TRACKS PER CYLINDER = 19,                                    00000950
        TRACK CAPACITY = (13030, CHARACTERS),                        00000960

........10........20........30........40........50........60........70........80
```

INPUT SEQ-NO	ALT INPUT LEV	ALT INPUT SEQ-NO	MEMBER REF-NO
73			60
74			61
75			62
76			63
77			64
78			65
79			66
80			67
81			68
82			69
83			
84			70
85			71
86			72
87			
88			73
89			74
90			
91			75
92			76
93			77
94			78
95			79
96			80

```
                                INPUT CARD IMAGE
........10........20........30........40........50........60........70........80
```

	INPUT SEQ-NO	ALT INPUT LEV	ALT INPUT SEQ-NO	MEMBER REF-NO
ROTATIONAL SPEED = 3600,	00000970	97		81
TRANSFER RATE = 806000,	00000980	98		82
SPACE OVERHEAD = 0,	00000990	99		83
CYLINDER ACCESS = PROC(CYL333);	00001000	100		84
	00001010	101		
CONTROL UNIT: ID = CU2804,	00001020	102		85
MAX TRANSFER RATE = 120000;	00001030	103		86
	00001040	104		
ACCESS MECHANISM: ID = (TP2401,6),	00001050	105		87
VOLUME = VL2401(1,6),	00001060	106		88
DEVICE = DV2401;	00001070	107		89
	00001080	108		
VOLUME: ID = (VL2401,6),	00001090	109		90
DEVICE = DV2401;	00001100	110		91
	00001110	111		
DEVICE: ID = DV2401,	00001120	112		92
TYPE = TAPE,	00001130	113		93
DENSITY = 1600,	00001140	114		94
SPEED = 75,	00001150	115		95
IRG = .6,	00001160	116		96
START STOP TIME = 13,	00001170	117		97
BLOCKSIZE = (VARIABLE, MAX(32767)),	00001180	118		98
FORWARD ERASE LENGTH = 3,	00001190	119		99
REWIND RATE = 350;	00001200	120		100

```
........10........20........30........40........50........60........70........80
```

```
                                    INPUT CARD IMAGE
........10........20........30........40........50........60........70........80     INPUT SEQ-NO  ALT INPUT LEV SEQ-NO  MEMBER REF-NO

                                                                                00001210      121
CONTROL UNIT: ID = CU2821,                                                      00001220      122                  101
              MAX TRANSFER RATE = 200000;                                       00001230      123                  102
                                                                                00001240      124
ACCESS MECHANISM: ID = CR2540,                                                  00001250      125                  103
                  DEVICE = DV254R;                                              00001260      126                  104
                                                                                00001270      127
DEVICE: ID = DV254R,                                                            00001280      128                  105
        TYPE = UR,                                                              00001290      129                  106
        CYCLE TIME = 60,                                                        00001300      130                  107
        BLOCKSIZE = (VARIABLE, MAX(80)),                                        00001310      131                  108
        MODE = SYNCHRONOUS,                                                     00001320      132                  109
        BUFFERED = YES;                                                         00001330      133                  110
                                                                                00001340      134
ACCESS MECHANISM: ID = CR2540,                                                  00001350      135                  111
                  DEVICE = DV254P;                                              00001360      136                  112
                                                                                00001370      137
DEVICE: ID = DV254P,                                                            00001380      138                  113
        TYPE = UR,                                                              00001390      139                  114
        CYCLE TIME = 200,                                                       00001400      140                  115
        BLOCKSIZE = (VARIABLE, MAX(80)),                                        00001410      141                  116
        MODE = SYNCHRONOUS,                                                     00001420      142                  117
        BUFFERED = YES;                                                         00001430      143                  118
                                                                                00001440      144

........10........20........30........40........50........60........70........80
```

```
                    INPUT CARD IMAGE
.......10........20........30........40........50........60........70........80
```

	INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
ACCESS MECHANISM: ID = PR1403,	00001450	145	119
DEVICE = DV1403;	00001460	146	120
	00001470	147	
DEVICE: ID = DV1403,	00001480	148	121
TYPE = UR,	00001490	149	122
BLOCKSIZE = (VARIABLE, MAX(133)),	00001500	150	123
MODE = SYNCHRONOUS,	00001510	151	124
CYCLE TIME = 54.5;	00001520	152	125
	00001530	153	
VOLUME: ID=FCONV, DEVICE=DV7100;	00001540	154	126
ACCESS MECHANISM: ID = CONSOL,	00001550	155	127
VOLUME = FCONV,	00001560	156	128
DEVICE = DV7100;	00001570	157	129
	00001580	158	
DEVICE: ID = DV7100,	00001590	159	130
TYPE = TAPE,	00001600	160	131
BLOCKSIZE = (VARIABLE, MAX(133)),	00001610	161	132
DENSITY = 8,	00001620	162	133
SPEED = 960,	00001630	163	134
IBG = .6,	00001640	164	135
START STOP TIME = 1.,	00001650	165	136
FORWARD ERASE LENGTH = 3,	00001660	166	137
REWIND RATE = 150;	00001670	167	138
	00001680	168	

```
.......10........20........30........40........50........60........70........80
```

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```
                              INPUT CARD IMAGE                                       INPUT    ALT INPUT   MEMBER
........10........20........30........40........50........60........70........80     SEQ-NO  LEV SEQ-NO   REF-NO

                                                                      00001681        169
VOLUME:  ID=CONV, DEVICE=DV1052:                                      00001682        170                  139
ACCESS MECHANISM: ID = CN1052,                                        00001683        171                  140
                  VOLUME = CONV,                                      00001684        172                  141
                  DEVICE = DV1052:                                    00001685        173                  142
                                                                      00001686        174
DEVICE: ID = DV1052,                                                  00001687        175                  143
        TYPE = TAPE,                                                  00001688        176                  144
        BLOCKSIZE = (VARIABLE, MAX(133)),                             00001689        177                  145
        DENSITY = 8,                                                  00001690        178                  146
        SPEED = 11,                                                   00001691        179                  147
        IBG = .6,                                                     00001692        180                  148
        START STOP TIME = 500.,                                       00001693        181                  149
        FORWARD ERASE LENGTH = 3,                                     00001694        182                  150
        REWIND RATE = 350:                                            00001695        183                  151
                                                                      00001696        184
DATA SET: ID = DDS1:                                                  00001697        185                  152
DATA SET: ID = TDS1:                                                  00001700        186                  153
DATA SET: ID = CDS1:                                                  00001710        187                  154
PROCEDURE: NAME = CPUTME,                                             00001720        188                  155
           TYPE = SUBROUTINE,                                         00001730        189                  156
           PARAMETER LIST = (MCTME):                                  00001740        190                  157
      REAL MCTME                                                      00001750        191                  158
C *** IBM 360 MODEL 30  RETURN MEMORY CYCLE TIME IN MICRO SECONDS.     00001760        192                  159

........10........20........30........40........50........60........70........80
```

```
                                  INPUT CARD IMAGE
........10........20........30........40........50........60........70........80
      MCTMF = 1.5                                                    00001770
      RETURN                                                         00001780
 END: PROCEDURE;                                                     00001790
                                                                     00001800
 PROCEDURE: NAME = CYLACC,                                           00001810
            TYPE = SUBROUTINE;                                       00001820
                                                                     00001830
      REAL CYLS, $PWLIN                                              00001840
      REAL POINTS(4,2) /25., 65., 75., 135., 0., 20., 80., 200./     00001850
                                                                     00001860
C                                                                    00001870
      CYLS = IABS(SYSCOM(2) - SYSCOM(1))                             00001880
C                                                                    00001890
      CALL $SRL1($PWLIN(POINTS,4,CYLS), SYSCOM(3))                   00001900
      RETURN                                                         00001910
 END: PROCEDURE;                                                     00001914
                                                                     00001919
 PROCEDURE: NAME = CYL333,                                           00001920
            TYPE = SUBROUTINE;                                       00001930
                                                                     00001940
      REAL CYLS, $PWLIN                                              00001950
      REAL POINTS(2,2) /10., 55., 0.0, 404./                         00001960
                                                                     00001970
C                                                                    00001980
........10........20........30........40........50........60........70........80
```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
193		160
194		161
195		162
196		
197		163
198		164
199		
200		166
201		167
202		
203		169
204		170
205		171
206		172
207		173
208		174
209		
210		175
211		176
212		
213		178
214		179
215		
216		181

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```
                    INPUT CARD IMAGE                                               INPUT    ALT INPUT   MEMBER
........10........20........30........40........50........60........70........80    SEQ-NO   LEV SEQ-NO  REF-NO

      CYLS = IABS(SYSCOM(2) - SYSCOM(1))                             00001990   217                182
C                                                                    00002000   218                183
      CALL $SRLI($PWLIN(POINTS,2,CYLS), SYSCOM(3))                   00002010   219                184
      RETURN                                                         00002020   220                185
C************************************************************        00002030   221                186
 END: PROCEDURE;                                                     00002040   222                187
                                                                     00002050   223
                                                                     00002060   224
                                                                     00002070   225
      EXC SERVICE: ID=INIT, NAME=INITS;                              00000010   226                188
      COMMON /BP/ BPOOL(5,50,4)                                      00000020   227                189
      INTEGER I,BPCOL,J,K,RETCU,RETCS                                00000030   228                190
                                                                     00000040   229
      END: DECLARATIONS;                                             00000050   230                192
      END: INITIALIZATION;                                           00000060   231                193
C                                                                    00000070   232                194
C *** GET FT05 PARAMETER AND FT01,FT02 FILE TABLES                   00000080   233                195
C                                                                    00000090   234                196
CTRACEWRITE(4,5)                                                     00000100   235                197
    5 FORMAT(' ','**MSG  INIT BEFORE CALL RSUF')                     00000110   236                198
      CALL RSUF(RETCU,RETCS)                                         00000120   237                199
CTRACEWRITE(4,15) RETCU,RETCS                                        00000130   238                200
   15 FORMAT(' ','***MSG INIT AFTER CALL RSUF-RETCU&CS- ',           00000140   239                201
     1       2(I2,1X))                                               00000150   240                202

........10........20........30........40........50........60........70........80
```

```
                                INPUT CARD IMAGE
........10........20........30........40........50........60........70........80     INPUT SEQ-NO   ALT INPUT LEV SEQ-NO   MEMBER REF-NO

C                                                                                    00000160        241                    203
      DC 20 I = 1,5                                                                  00000170        242                    204
        DO 20 J = 1,50                                                               00000180        243                    205
          DO 20 K = 1,4                                                              00000190        244                    206
   20 BPOOL(I,J,K) = 0                                                             00000200        245                    207
C     IF(PETCU .NE. 0) GO TO 100                                                     00000210        246                    208
C     IF(PETCS .NE. 0) GO TO 100                                                     00000220        247                    209
C *** CREATE ALL DATA SETS                                                           00000230        248                    210
C                                                                                    00000240        249                    211
      CREATE DATASET: DATASET = DDS1, INDEX=1;                                       00000250        250                    212
      ALLOCATE DATASET EXTENT: DATASET = DDS1, EXTENT = 1;                           00000260        251                    213
C                                                                                    00000270        252                    214
      CREATE DATASET: DATASET = TDS1, INDEX = 1;                                     00000280        253                    215
      ALLOCATE DATA SET EXTENT: DATASET = TDS1, EXTENT = 1;                          00000290        254                    216
C                                                                                    00000300        255                    217
CTRACEWRITE(4,25)                                                                    00000310        256                    218
   25 FORMAT(' ','***MSG INIT AFTER FILE CREATES')                                   00000320        257                    219
      CREATE DATASET: DATASET = CDS1, INDEX = 1;                                     00000330        258                    220
      ALLOCATE DATA SET EXTENT: DATASET = CDS1, EXTENT = 1;                          00000340        259                    221
  100 CONTINUE                                                                       00000350        260                    222
C ***************************************************************                  00000360        261                    223
      END; EXO SERVICE;                                                              00000370        262                    224
                                                                                     00000380        263
                                                                                     00000390        264

........10........20........30........40........50........60........70........80
```

```
                                INPUT CARD IMAGE
........10........20........30........40........50........60........70........80
      EXD SERVICE: ID = START, NAME = STARTX,                          00000400
                   SAVE AREA SIZE = 30;                               00000410
C                                                                     00000420
      INTEGER I,ISEED,KSEED,NAPS, NSFL, NUFL, NUM                     00000430
      EQUIVALENCE (I,$SAVE(1)),(NUM,$SAVE(2))                         00000440
      COMMON /USERD/ NAPS, NUFL, NSFL                                 00000450
      COMMON /SEED/ ISEED                                             00000460
C                                                                     00000470
      END: DECLARATIONS;                                              00000480
      END: INITIALIZATION;                                            00000490
C                                                                     00000500
      SEIZE: FACILITY = START;                                        00000510
      NUM = NAPS                                                      00000520
      KSEED = ISEED                                                   00000530
                                                                      00000540
      I = 0                                                           00000550
    5 I = I + 1                                                       00000560
CTRACEWRITE(6,12) I,NUM                                               00000570
   12 FORMAT(' *,'***MSG START -I,NUM- ',2(I6,2X))                    00000580
      IF (I .GT. NUM) GO TO 20                                        00000590
      INVOKE: SERVICE = APPLX,                                        00000600
              PARAMETER LIST = (I);                                   00000610
      GO TO 5                                                         00000620
   20 CONTINUE                                                        00000630
........10........20........30........40........50........60........70........80
```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
265		225
266		226
267		227
268		228
269		229
270		230
271		231
272		232
273		233
274		234
275		235
276		236
277		237
278		238
279		
280		240
281		241
282		242
283		243
284		244
285		245
286		246
287		247
288		248

```
                    INPUT CARD IMAGE
.......10.......20.......30.......40.......50.......60.......70.......80
CTRACEWRITE(4,14)                                                        00000640   289                249
   14 FORMAT(' ','***MSG START AFTER 20 AND BEFORE WAIT APPL')           00000650   290                250
      WAIT RETURN: SERVICE = APPL;                                       00000660   291                251
      INVOKE: SERVICE = BUFSTX;                                          00000670   292                252
      WAIT RETURN: SERVICE = BUFST;                                      00000680   293                253
CTRACEWRITE(4,24) NUM                                                    00000690   294                254
   24 FORMAT(' ','***MSG START AFTER WAIT BUFST -NUM- ',I6)               00000700   295                255
      IF (NUM .LE. 1) GO TO 90                                           00000710   296                256
      NUM = NUM - 1                                                      00000720   297                257
      GO TO 20                                                           00000730   298                258
   90 CONTINUE                                                           00000740   299                259
      WRITE(6,2) KSEED, ISEED                                            00000750   300                260
2     FORMAT(//,20X, 'BEGINNING SEED FOR R.N.G. - ',I10,                 00000760   301                261
     +       /,20X, '   ENDING SEED FOR R.N.G. - ',I10)                 00000770   302                262
      RELEASE: FACILITY = START;                                         00000780   303                263
C *************************************************************          00000790   304                264
      END: EXD SERVICE;                                                  00000800   305                265
                                                                         00000810   306
                                                                         00000820   307
      ENDD SERVICE: ID = APPL,                                           00000830   308                266
                    NAME = APPLX,                                        00000840   309                267
                    SAVE AREA SIZE = 30,                                 00000850   310                268
                    PARAMETER LIST = (APPLX);                            00000860   311                269
C                                                                        00000870   312                270
.......10.......20.......30.......40.......50.......60.......70.......80
```

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```
                                  INPUT CARD IMAGE                                  INPUT     ALT INPUT  MEMBER
.......10........20........30........40........50........60........70........80     SEQ-NO   LEV SEQ-NO  REF-NO

C                                                                                 00000880      313          271
C *** THIS SERVICE REPRESENTS APPLICATION PROCESSING.   EXECUTION                 00000890      314          272
C *** GROUPS ARE READ IN (ONE AT A TIME) AND A SEQUENCE OF READS AND              00000900      315          273
C *** WRITES IS ISSUED TO THE NEXT (LOWER) SERVICE                                00000910      316          274
C                                                                                 00000920      317          275
C     THIS SERVICE                                                                00000930      318          276
C        CALLS RDEG TO READ ALL PROCESSING GROUPS OF                              00000940      319          277
C                  OF AN EXECUTION GROUP, AND                                     00000950      320          278
C        CALLS EXGP TO PERFORM THE I/O FOR EACH EXECUTION GROUP                   00000960      321          279
C                                                                                 00000970      322          280
C     *APPLICATION PROCESSING TABLE*                                              00000980      323          281
C     NUMERIC I/O TABLE - NIOT                                                    00000990      324          282
C        (APS,I,1)                                                                00001000      325          283
C        (APS,I,2)                                                                00001010      326          284
C        (APS,I,3) - INDEX OF FILE IN FILET                                       00001020      327          285
C        (APS,I,4) - POINTER TO SYSFT                                             00001030      328          286
C        (APS,I,5) - NUMBER OF APPLICATION RECORDS READ OR WRITTEN                00001040      329          287
C        (APS,I,6)                                                                00001050      330          288
C                                                                                 00001060      331          289
C                                                                                 00001070      332          290
      INTEGER APS,APST(5),APSX,IOPT(20),RETC,UNIT                                 00001080      333          291
      END DECLARATIONS;                                                           00001090      334          292
      APS = APSX                                                                  00001100      335          293
C     GET PROCESSING GROUP FROM EACH APPLICATION                                  00001110      336          294

.......10........20........30........40........50........60........70........80
```

```
                                    INPUT CARD IMAGE
.......10........20........30........40........50........60........70........80     INPUT SEQ-NO  ALT INPUT LEV SEQ-NO  MEMBER REF-NO

      END: INITIALIZATION:                                                        00001120        337                    295
      SEIZE: FACILITY = APPL;                                                     00001130        338                    296
      WRITE(6,7) APS                                                              00001140        339                    297
7     FORMAT(1H1,15X,'I/O PROCESSING TABLE FOR APPLICATION ', I2)                 00001150        340                    298
      UNIT = 9 + APS                                                              00001160        341                    299
3     READ(UNIT,1,END=4)  (IOPT(J), J=1,20)                                       00001170        342                    300
1     FORMAT(20A4)                                                                00001180        343                    301
      WRITE(6,2)  (IOPT(J), J=1,20)                                               00001190        344                    302
2     FORMAT(/,10X,20A4)                                                          00001200        345                    303
      GO TO 3                                                                     00001210        346                    304
4     REWIND UNIT                                                                 00001220        347                    305
C        BEGIN EXECUTION GROUP PROCESSING HERE                                    00001230        348                    306
    5 CONTINUE                                                                    00001240        349                    307
CTRACEWRITE(4,6) APS                                                              00001250        350                    308
    6 FORMAT(' ','***MSG APPL AT BEGINNING -APS- ',I6)                            00001260        351                    309
      CALL RDGP(APS,RETC)                                                         00001270        352                    310
      APST(APS) = RETC                                                            00001280        353                    311
C     RETC = 1  OK                                                                00001290        354                    312
C            2  EOF - NO DATA TO BE PROCESSED HERE                                00001300        355                    313
C            3  BAD DATA - DON'T PROCESS IT HERE                                  00001310        356                    314
C            4  TOO MUCH DATA - DON'T PROCESS IT HERE                             00001320        357                    315
      IF(RETC .LE. 1) GO TO 8                                                     00001330        358                    316
      IF(RETC .EQ. 2) GO TO 900                                                   00001340        359                    317
C     BAD DATA OR TOO MUCH DATA - GET NEXT EXEC GROUP                             00001350        360                    318

.......10........20........30........40........50........60........70........80
```

```
                                  INPUT CARD IMAGE
........10........20........30........40........50........60........70........80
      GO TO 5                                                      00001360
C     PROCESS EACH PROCESSING GROUP OF THIS EXECUTION GROUP        00001370
    8 CONTINUE                                                     00001380
      INVOKE: SERVICE = EXGPX,                                     00001390
              PARAMETER LIST = (APS);                              00001400
      WAIT RETURN: SERVICE = EXGP;                                 00001410
C     PRINT STATISTICS AFTER COMPLETION OF EACH EXECUTION GROUP    00001420
      CALL $SNAP                                                   00001430
CTRACEWRITE(4,2)                                                   00001440
                                                                   00001450
      GO TO 5                                                      00001460
  900 CONTINUE                                                     00001470
      RELEASE: FACILITY = APPL;                                    00001480
C                                                                  00001490
C ***************************************************************** 00001500
  END: ENDO SERVICE;                                               00001510
                                                                   00001520
                                                                   00001530
                                                                   00001540
      ENDO SERVICE: ID = EXGP,                                     00001550
                    NAME = EXGPX,                                  00001560
                    SAVE AREA SIZE = 30,                           00001570
                    PARAMETER LIST = (APSX);                       00001580
      COMMON /USERP/ NAPS, NUFL, NSFL                              00001590

........10........20........30........40........50........60........70........80
```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
361		319
362		320
363		321
364		322
365		323
366		324
367		325
368		326
369		327
370		
371		329
372		330
373		331
374		332
375		333
376		334
377		
378		
379		
380		335
381		336
382		337
383		338
384		339

```
                                INPUT CARD IMAGE
.....10.....20.....30.....40.....50.....60.....70.....80     INPUT SEQ-NO  ALT INPUT LEV SEQ-NO  MEMBER REF-NO
C     3       FACTYP,FDD,FDL,FEDD,FID,I,LREC,LRECL,                00004520      1537                  1474
C     4       MDS,NBPPR,NBPRU,NEXT,NF,NSEG,PREC,PRECL,REC,         00004530      1538                  1475
C     5       RPREC,RUTYPE,S,SDD,ST,TC,TPC,TRK,VOL,                00004540      1539                  1476
C     6       EMPT,ISFT,LRECLX,NRECS,NSEXT,PPEXT,PDRSX,UFNO,       00004550      1540                  1477
C     7       VOLNUM,VOLTYP,                                       00004560      1541                  1478
C     8       ESIZE,NRU,PRDA,PRDE,PRDS,PRDV,PRDVC,SDV,             00004570      1542                  1479
C     9       SSIZE,T,X)                                           00004580      1543                  1480
C       AT 10                                                      00004590      1544                  1481
C       TRACE ON                                                   00004600      1545                  1482
C*************************************************************    00004610      1546                  1483
   END: PROCEDURE;                                                 00004620      1547                  1484
                                                                   00004630      1548
                                                                   00004640      1549
                                                                   00004650      1550
      ENDO SERVICE: ID=CHPGM,                                      00000010      1551                  1485
                    NAME = CHPGMX,                                 00000020      1552                  1486
                    SAVE AREA SIZE = 30,                           00000030      1553                  1487
                    PARAMETER LIST = (DSX,FX,LRECX,ISFTX,RDWTX);   00000040      1554                  1488
                                                                   00000050      1555
      INTEGER AM,AVAIL,CHSTAT(2),CYL,DS,DSX,DVTYPE,F,FX            00000060      1556                  1490
      INTEGER IODATA,INUSE,ISFT,ISFTX,LREC,LRECX,PARM,PRECL        00000070      1557                  1491
      INTEGER RDWT,RDWTX,SELSW,VOL                                 00000080      1558                  1492
      DATA AVAIL/0/, INUSE/1/, CHSTAT/0,0/                         00000090      1559                  1493
      DIMENSION PARM(5),IODATA(15)                                 00000100      1560                  1494
.....10.....20.....30.....40.....50.....60.....70.....80
```

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```
                         INPUT CARD IMAGE                                      INPUT   ALT INPUT  MEMBER
........10........20........30........40........50........60........70........80   SEQ-NO  LEV SEQ-NO REF-NO

      EQUIVALENCE (PARM(1),DS), (PARM(2),F), (PARM(3),LREC),            00000110   1561              1495
     +            (PARM(4),ISFT), (PARM(5),RDWT),                       00000120   1562              1496
     +            (VOL,IODATA(2)), (AM,IODATA(3)), (DVTYPE,IODATA(5)),  00000130   1563              1497
     +            (CFCV,IODATA(4)),                                     00000140   1564              1498
     +            (CYL,IODATA(6)), (PRECL,IODATA(11)),                  00000150   1565              1499
     +            (PARM,$SAVE(16)), (IODATA,$SAVE)                      00000160   1566              1500
C                                                                       00000170   1567              1501
      END: DECLARATIONS:                                                00000180   1568              1502
C                                                                       00000190   1569              1503
C *** TRANSFER PARAMETERS TO REGULAR FORTRAN VARIABLES                  00000200   1570              1504
C                                                                       00000210   1571              1505
      DS = DSX                                                          00000220   1572              1506
      F = FX                                                            00000230   1573              1507
      LREC = LRECX                                                      00000240   1574              1508
      ISFT = ISFTX                                                      00000250   1575              1509
      RDWT = RDWTX                                                      00000260   1576              1510
C                                                                       00000270   1577              1511
      END: INITIALIZATION:                                              00000280   1578              1512
      SEIZE: FACILITY = CHPGM;                                          00000290   1579              1513
C                                                                       00000300   1580              1514
C *** GET THE PHYSICAL ADDRESS OF THE LOGICAL RECORD TO BE USED         00000310   1581              1515
C                                                                       00000320   1582              1516
CTRACEWRITE(4,14) DS,F,LREC,ISFT                                        00000330   1583              1517
   14 FORMAT(' ','***MSG CHPGM BEFORE GADRU -DS,F,LREC,ISFT-',          00000340   1584              1518

........10........20........30........40........50........60........70........80
```

```
                                INPUT CARD IMAGE
.....10.....20.....30.....40.....50.....60.....70.....80     INPUT SEQ-NO  ALT INPUT LEV SEQ-NO  MEMBER REF-NO
     1       4(I5,2X))                                        00000350      1585                  1519
      CALL GADRU(DS,F,LREC,ISFT,IODATA)                       00000360      1586                  1520
CTRACEWRITE(4,10) IODATA                                      00000370      1587                  1521
   10 FORMAT(' ',' ***MSG CHPGM AFTER GADRU -IODATA-',        00000380      1588                  1522
     1       15(I5,1X))                                       00000390      1589                  1523
C                                                             00000400      1590                  1524
C *** ACQUIRE THE FACILITIES TO EFFECT THE DATA TRANSFER. FIRST 00000410     1591                  1525
C *** DETERMINE IF I/O IS TAPE OR DISK                        00000420      1592                  1526
C                                                             00000430      1593                  1527
      IF (DVTYPE .NE. 1) GO TO 200                            00000440      1594                  1528
C                                                             00000450      1595                  1529
C *** DISK I/O, ACQUIRE FACILITIES IN REVERSE ORDER           00000460      1596                  1530
C                                                             00000470      1597                  1531
      QUEUE: FACILITY = AM;                                   00000480      1598                  1532
      WAIT FACILITY: FACILITY = AM,                           00000490      1599                  1533
                     STATUS = AVAIL;                          00000500      1600                  1534
      DEPART QUEUE: FACILITY = AM;                            00000510      1601                  1535
      SET STATUS: FACILITY = AM,                              00000520      1602                  1536
                  STATUS = INUSE;                             00000530      1603                  1537
      SEIZE: FACILITY = AM;                                   00000540      1604                  1538
C                                                             00000550      1605                  1539
      QUEUE: FACILITY = CU2314;                               00000560      1606                  1540
      WAIT FACILITY: FACILITY = CU2314,                       00000570      1607                  1541
                     STATUS = AVAIL;                          00000580      1608                  1542
.....10.....20.....30.....40.....50.....60.....70.....80
```

```
                                INPUT CARD IMAGE                                  INPUT    ALT INPUT  MEMBER
.....10.....20.....30.....40.....50.....60.....70.....80                          SEQ-NO   LEV SEQ-NO REF-NO

      DEPART QUEUE: FACILITY = CU2314;                                  00000590   1609               1543
      SET STATUS: FACILITY = CU2314,                                    00000600   1610               1544
                  STATUS = INUSE;                                       00000610   1611               1545
      SEIZE: FACILITY = CU2314;                                         00000620   1612               1546
C                                                                       00000630   1613               1547
      QUEUE: FACILITY = SELECT(1);                                      00000640   1614               1548
      WAIT FACILITY: FACILITY = SELECT(1),                              00000650   1615               1549
                     STATUS = AVAIL;                                    00000660   1616               1550
      DEPART QUEUE: FACILITY = SELECT(1);                               00000670   1617               1551
C                                                                       00000680   1618               1552
C *** THE CHANNEL IS ACQUIRED MOMENTARILY TO PERFORM SEEK               00000690   1619               1553
C                                                                       00000700   1620               1554
      SEEK: ACCESS MECHANISM = AM,                                      00000710   1621               1555
            CYLINDER = CYL;                                             00000720   1622               1556
      WAIT INPUT OUTPUT: IOTYPE = SEEK;                                 00000730   1623               1557
C                                                                       00000740   1624               1558
C *** SEEK IS COMPLETE, RE-ACQUIRE THE CHANNEL AND TRANSFER THE DATA     00000750   1625               1559
C                                                                       00000760   1626               1560
CTRACEWRITE(4,131)                                                      00000770   1627               1561
  131 FORMAT(' ',' ***MSG CHPGM AFTER SEEK IS DONE ')                   00000780   1628               1562
      QUEUE: FACILITY = SELECT(1);                                      00000790   1629               1563
      WAIT FACILITY: FACILITY = SELECT(1),                              00000800   1630               1564
                     STATUS = AVAIL;                                    00000810   1631               1565
      DEPART QUEUE: FACILITY = SELECT(1);                               00000820   1632               1566

.....10.....20.....30.....40.....50.....60.....70.....80
```

```
                                INPUT CARD IMAGE                                 INPUT    ALT INPUT  MEMBER
.....10.....20.....30.....40.....50.....60.....70.....80                         SEQ-NO   LEV SEQ-NO REF-NO

      SET STATUS: FACILITY = SELECT(1),                                          00000830   1633       1567
                  STATUS = INUSE;                                                00000840   1634       1568
      CHSTAT(1) = INUSE                                                          00000850   1635       1569
      SEIZE: FACILITY = SELECT(1);                                               00000860   1636       1570
C                                                                                00000870   1637       1571
C *** TEST THE TYPE OF I/O TRANSFER                                              00000880   1638       1572
C                                                                                00000890   1639       1573
      IF (RDWT .NE. 1) GO TO 150                                                 00000900   1640       1574
      DATA TRANSFER: TYPE = READ,                                                00000910   1641       1575
                     LATENCY = YES,                                              00000920   1642       1576
                     VOLUME = VOL,                                               00000930   1643       1577
                     DATASET = DS.F,                                             00000940   1644       1578
                     PHYSICAL RECORD LENGTH = PRECL;                             00000950   1645       1579
       GO TO 175                                                                 00000960   1646       1580
  150 CONTINUE                                                                   00000970   1647       1581
      DATA TRANSFER: TYPE = WRITE,                                               00000980   1648       1582
                     LATENCY = YES,                                              00000990   1649       1583
                     VOLUME = VOL,                                               00001000   1650       1584
                     DATASET = DS.F,                                             00001010   1651       1585
                     PHYSICAL RECORD LENGTH = PRECL;                             00001020   1652       1586
  175 CONTINUE                                                                   00001030   1653       1587
      WAIT INPUT OUTPUT: IOTYPE = DATA TRANSFER;                                 00001040   1654       1588
C                                                                                00001050   1655       1589
C *** RELEASE ALL OF THE HELD FACILITIES AND SPLIT                               00001060   1656       1590

.....10.....20.....30.....40.....50.....60.....70.....80
```

```
                                INPUT CARD IMAGE
........10........20........30........40........50........60........70........80
C                                                                               00001070
      RELEASE: FACILITY = SELECT(1);                                            00001080
      SET STATUS: FACILITY = SELECT(1),                                         00001090
                  STATUS = AVAIL;                                               00001100
      CHSTAT(1) = AVAIL                                                         00001110
      RELEASE: FACILITY = CU2314;                                               00001120
      SET STATUS: FACILITY = CU2314,                                            00001130
                  STATUS = AVAIL;                                               00001140
      RELEASE: FACILITY = AM;                                                   00001150
      SET STATUS: FACILITY = AM,                                                00001160
                  STATUS = AVAIL;                                               00001170
      GO TO 900                                                                 00001180
C                                                                               00001190
C *** NOW HANDLE TAPE I/O                                                       00001200
C                                                                               00001210
  200 IF (DVTYPE .NE. 2) GO TO 800                                              00001220
      QUEUE: FACILITY = AM;                                                     00001230
      WAIT FACILITY: FACILITY = AM,                                             00001240
                     STATUS = AVAIL;                                            00001250
      DEPART QUEUE: FACILITY = AM;                                              00001260
      SET STATUS: FACILITY = AM,                                                00001270
                  STATUS = INUSE;                                               00001280
      SEIZE: FACILITY = AM;                                                     00001290
C                                                                               00001300
........10........20........30........40........50........60........70........80
```

INPUT SEQ-NO	ALT INPUT LEV SEQ-NO	MEMBER REF-NO
1657		1591
1658		1592
1659		1593
1660		1594
1661		1595
1662		1596
1663		1597
1664		1598
1665		1599
1666		1600
1667		1601
1668		1602
1669		1603
1670		1604
1671		1605
1672		1606
1673		1607
1674		1608
1675		1609
1676		1610
1677		1611
1678		1612
1679		1613
1680		1614

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```
                         INPUT CARD IMAGE                                            INPUT    ALT INPUT    MEMBER
.......10........20........30........40........50........60........70........80      SEQ-NO  LEV SEQ-NO   REF-NO

      QUEUE: FACILITY = CU2804;                                        00001310      1681                 1615
      WAIT FACILITY: FACILITY = CU2804,                                00001320      1682                 1616
                     STATUS = AVAIL;                                   00001330      1683                 1617
      DEPART QUEUE: FACILITY = CU2804;                                 00001340      1684                 1618
      SET STATUS: FACILITY = CU2804,                                   00001350      1685                 1619
                  STATUS = INUSE;                                      00001360      1686                 1620
      SEIZE: FACILITY = CU2804;                                        00001370      1687                 1621
C                                                                      00001380      1688                 1622
C *** CHECK CHANNELS TO SIMULATE DUAL CONTROLLER DOWN SWITCHING, IF     00001390      1689                 1623
C *** CHANNEL 2 IS BUSY WAIT FOR CHANNEL 1 TO FREE                      00001400      1690                 1624
C                                                                      00001410      1691                 1625
      SELSW = 2                                                        00001420      1692                 1626
      IF (CHSTAT(2) .EQ. INUSE) SELSW = 1                              00001430      1693                 1627
C                                                                      00001440      1694                 1628
C *** WAIT FOR CHANNEL TO FREE                                         00001450      1695                 1629
C                                                                      00001460      1696                 1630
      QUEUE: FACILITY = SELECT(SELSW);                                 00001470      1697                 1631
      WAIT FACILITY: FACILITY = SELECT(SELSW),                         00001480      1698                 1632
                     STATUS = AVAIL;                                   00001490      1699                 1633
      DEPART QUEUE: FACILITY = SELECT(SELSW);                          00001500      1700                 1634
      SET STATUS: FACILITY = SELECT(SELSW),                            00001510      1701                 1635
                  STATUS = INUSE;                                      00001520      1702                 1636
      CHSTAT(SELSW) = INUSE                                            00001530      1703                 1637
      SEIZE: FACILITY = SELECT(SELSW);                                 00001540      1704                 1638

.......10........20........30........40........50........60........70........80
```

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```
                          INPUT CARD IMAGE                                         INPUT    ALT INPUT   MEMBER
........10........20........30........40........50........60........70........80   SEQ-NO  LEV SEQ-NO  REF-NO

C                                                                       00001550   1705              1639
C *** TEST TYPE, IF TRANSFER THEN DO IT                                  00001560   1706              1640
C                                                                       00001570   1707              1641
      IF (RDWT .NE. 1) GO TO 350                                        00001580   1708              1642
      DATA TRANSFER: TYPE = READ,                                       00001590   1709              1643
                     DATA SET = DS.F.                                   00001600   1710              1644
                     VOLUME = VOL,                                      00001610   1711              1645
                     PHYSICAL RECORD LENGTH = PRECL;                    00001620   1712              1646
      GO TO 400                                                         00001630   1713              1647
  350 CONTINUE                                                          00001640   1714              1648
      DATA TRANSFER: TYPE = WRITE,                                      00001650   1715              1649
                     DATA SET = DS.F.                                   00001660   1716              1650
                     VOLUME = VOL,                                      00001670   1717              1651
                     PHYSICAL RECORD LENGTH = PRECL;                    00001680   1718              1652
C                                                                       00001690   1719              1653
C *** WAIT FOR COMPLETION AND RELEASE FACILITIES AND SPLIT               00001700   1720              1654
C                                                                       00001710   1721              1655
  400 CONTINUE                                                          00001720   1722              1656
      WAIT INPUT OUTPUT: IOTYPE = DATA TRANSFER;                        00001730   1723              1657
C                                                                       00001740   1724              1658
      RELEASE: FACILITY = SELECT(SELSW);                                00001750   1725              1659
      SET STATUS: FACILITY = SELECT(SELSW),                             00001760   1726              1660
                  STATUS = AVAIL;                                       00001770   1727              1661
      CHSTAT(SELSW) = AVAIL                                             00001780   1728              1662

........10........20........30........40........50........60........70........80
```

STANDARD INPUT STREAM LISTING

DATE ... 08/28/79

```
                         INPUT CARD IMAGE                                    INPUT   ALT INPUT  MEMBER
........10........20........30........40........50........60........70........80   SEQ-NO  LEV SEQ-NO  REF-NO

      RELEASE: FACILITY = CU2804;                                  00001790     1729              1663
      SET STATUS: FACILITY = CU2804,                               00001800     1730              1664
                 STATUS = AVAIL;                                   00001810     1731              1665
      RELEASE: FACILITY = AM;                                      00001820     1732              1666
      SET STATUS: FACILITY = AM,                                   00001830     1733              1667
                 STATUS = AVAIL;                                   00001840     1734              1668
      GO TO 900                                                    00001850     1735              1669
C                                                                  00001860     1736              1670
C                                                                  00001870     1737              1671
C     ***OK BILL HERE IT IS AND I THINK IT WILL WORK***            00001880     1738              1672
C                                                                  00001890     1739              1673
  800 IF(DVTYPE .NE. 4) GO TO 900                                  00001900     1740              1674
C     UNSPEC DEVICE TYPE                                           00001910     1741              1675
C     GET THE TRANSFER RATE FROM THE $CFDEV ARRAY                  00001920     1742              1676
      IVTYP = $CFDFF(CFDV+7)                                       00001930     1743              1677
      GO TO (810,812,814,816,818),IVTYP                            00001940     1744              1678
C     ERROR - PUT ASSUME THE TYPE IS INTEGER                       00001950     1745              1679
C                                                                  00001960     1746              1680
C     INTEGER IS SPECIFIED - ASSIGNMENT WILL CONVERT IT TO REAL    00001970     1747              1681
  810 TRATE = $CFFFF(CFDV+9)                                       00001980     1748              1682
      GO TO 820                                                    00001990     1749              1683
C     REAL IS SPECIFIED                                            00002000     1750              1684
  812 CALL $EPLI(TRATE,$CFDFF(CFDV+9))                             00002010     1751              1685
      GO TO 820                                                    00002020     1752              1686

........10........20........30........40........50........60........70........80
```

```
                        INPUT CARD IMAGE                                    INPUT    ALT INPUT  MEMBER
.......10........20........30........40........50........60........70........80    SEQ-NO   LEV SEQ-NO  REF-NO

C       SYSPAR VALUE IS SPECIFIED                                          00002030    1753                1687
  814   TRATE = SYSPAR($CFDEF(CFDV+8))                                    00002040    1754                1688
        GO TO 820                                                          00002050    1755                1689
C       PROCEDURE REFERENCE IS SPECIFIED                                   00002060    1756                1690
  816   CALL $CFPR($CFDEF(CFDV+8))                                         00002070    1757                1691
        TRATE = SYSCOM(1)                                                  00002080    1758                1692
        GO TO 820                                                          00002090    1759                1693
C       GLOBAL PROCEDURE REFERENCE IS SPECIFIED                            00002100    1760                1694
C       EPROP - GPROC IS NOT IMPLEMENTED                                   00002110    1761                1695
  818   TRATE = 10                                                         00002120    1762                1696
C       TRATE IS ASSUMED TO BE SPECIFIED IN CHARACTERS PER SECOND          00002130    1763                1697
C       COMPUTE THE TIME REQUIRED TO TRANSFER PRECL CHARACTERS (BYTES)     00002140    1764                1698
C         IN MSEC TIME UNITS                                               00002150    1765                1699
  820   TME = (PRECL/TRATE) * 1000                                         00002160    1766                1700
        QUEUE: FACILITY = AM;                                              00002170    1767                1701
        WAIT FACILITY: FACILITY = AM,                                      00002180    1768                1702
                       STATUS = AVAIL;                                     00002190    1769                1703
        DEPART QUEUE: FACILITY = AM;                                       00002200    1770                1704
        SET STATUS: FACILITY = AM,                                         00002210    1771                1705
                    STATUS = INUSE;                                        00002220    1772                1706
        SEIZE: FACILITY = AM;                                              00002230    1773                1707
C       HOLD THE AM FOR THE DURATION OF THE DATA TRANSFER                  00002240    1774                1708
C       NOTE - THE IPSS DATA TRANSFER STATEMENT ISN'T SUITABLE FOR         00002250    1775                1709
C        UNIT RECORD OR UNSPEC TYPE DEVICES                                00002260    1776                1710

.......10........20........30........40........50........60........70........80
```

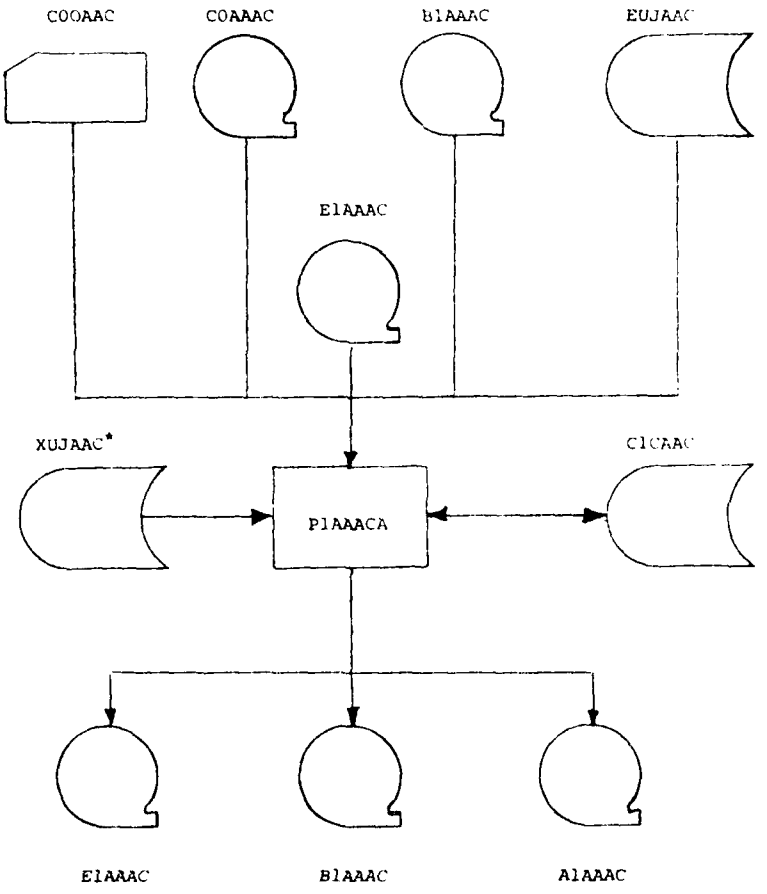
```
                                  INPUT CARD IMAGE                                  INPUT    ALT INPUT  MEMBER
.....10.....20.....30.....40.....50.....60.....70.....80                            SEQ-NO   LEV SEQ-NO REF-NO

      PROCESS:  TIME = TMF;                                                  00002270  1777               1711
      RELEASE: FACILITY = AM;                                                00002280  1778               1712
      SET STATUS: FACILITY = AM, STATUS = AVAIL;                             00002290  1779               1713
C                                                                            00002300  1780               1714
C *** ALL DONE, SO RELEASE THIS SERVICE AND SPLIT                            00002310  1781               1715
C                                                                            00002320  1782               1716
  900 CONTINUE                                                               00002330  1783               1717
      RELEASE: FACILITY = CHPGM;                                             00002340  1784               1718
C************************************************************               00002350  1785               1719
      END: ENDO SERVICE;                                                     00002360  1786               1720
                                                                             00002370  1787
                                                                             00002380  1788
                                                                             00002390  1789
                                                                             00000910  1790
                                                                             00000920  1791
      PROCEDURE: NAME = RSUF,                                                00000030  1792               1721
                 TYPE = SUBROUTINE,                                          00000940  1793               1722
                 PARAMETER LIST = (RETCU, RETCS);                            00000950  1794               1723
C                                                                            00000960  1795               1724
C        PROCEDURE RSUF READS AND PROCESSES INTO TABLES                      00000970  1796               1725
C        THE INPUT RECORDS DEFINING SYSTEM AND USER FILES                    00000980  1797               1726
C                                                                            00000990  1798               1727
      INTEGER BLANK, CLKSZ,CMT(15),DIMSET,DIMUFL,DTPEU(7),ETYPE,FILE         00001000  1799               1728
      INTEGER FILET,HCY,HTBL,I,INDEX,J,K,KNOWN,KU,LCYL                       00001110  1800               1729

.....10.....20.....30.....40.....50.....60.....70.....80
```

APPENDIX C

SIDPERS JOBSTEPS P1A THROUGH P1G

PROCESSING CHARACTERISTICS

This appendix contains the basic system flow charts for SIDPERS programs P1A, P1B, P1C, and P1G (Figures C-1 through C-4 respectively). For each program, Table C-1 presents a brief file description, record length and blocking factor specifications, the type of storage media on which the file resides, and an indicator of file use (input to the program, output from the program, or both input and output).



* X = A,B,C,E,F,G,H

Figure C-1. P1A File Identification

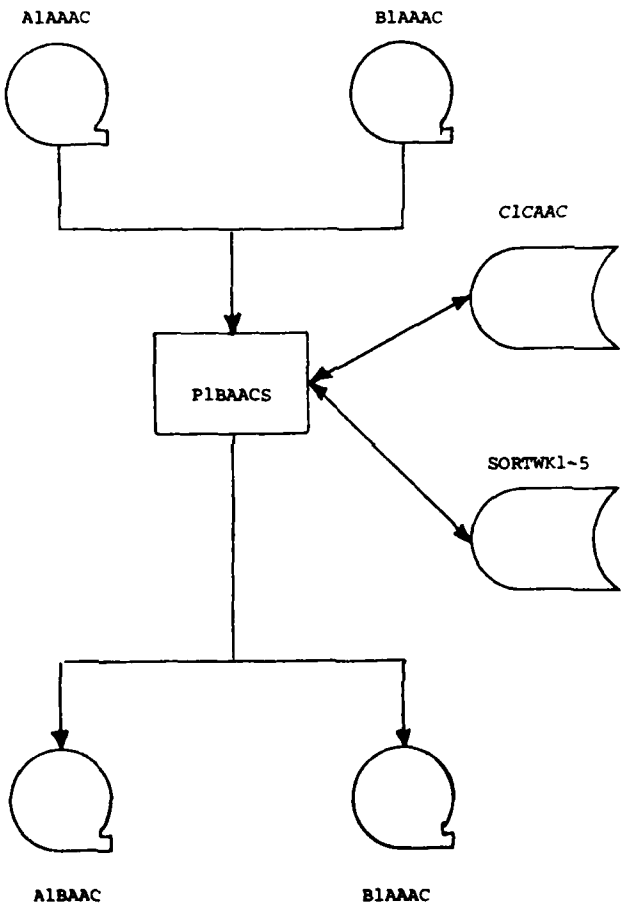


Figure C-2. P1B File Identification

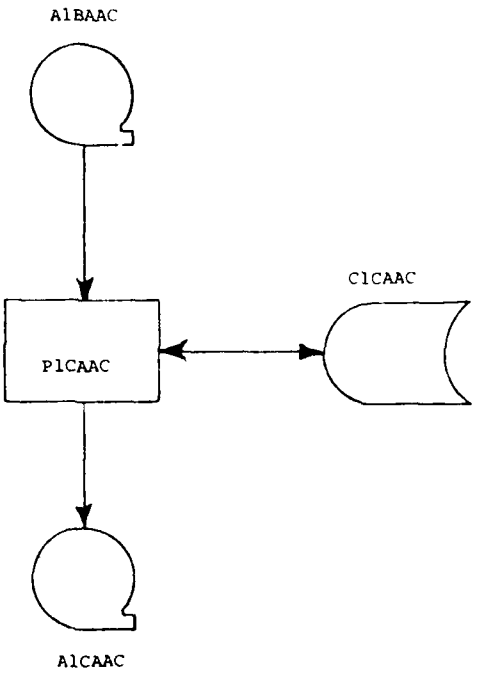
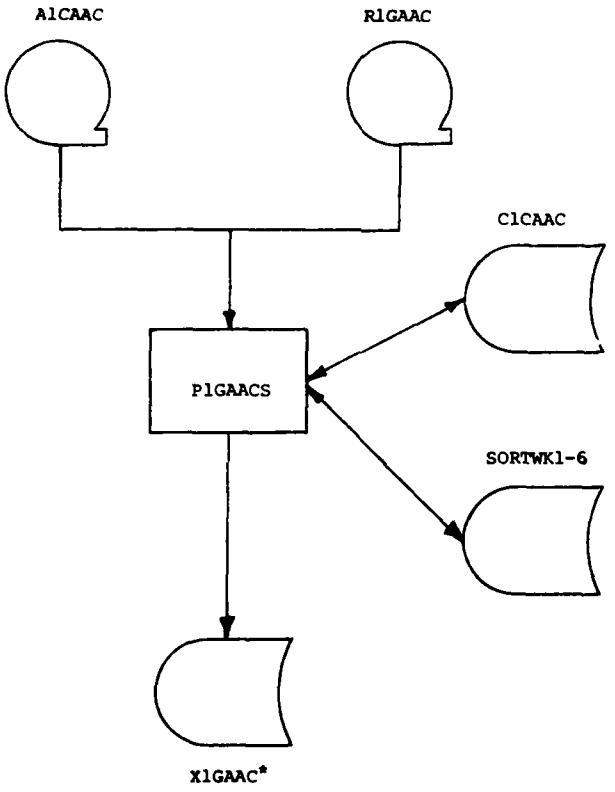


Figure C-3. P1C File Identification



* X = A,B,C,E,F,G,I,J,K,M,N,Q,R

Figure C-4. P1G File Identification

Table C-1. File Characteristics for Programs P1A Through P1C

File Name	Media	Input/ Output	Description	Logical Record Length	Blocking Factor
P1A					
C00AAC	Card/Tape	I	Optional Input	80	1
B1AAAC	Tape	I	Input Stacker File	100	40
C0AAAC	Tape	I	Transaction	80	1
A1AAAC	Tape	O	Class-sched trans file	132	10
E1AAAC	Tape	O	SIDPERS trans history	80	10
B1AAAC	Tape	O	Output Stacker file	100	40
C1CAAC	Disk	I/O	Edit table file	506	2
XUJAAC (X=A,B,C, E,F,G,H)	Disk	I		80	1
P1B					
A1AAAC	Tape	I	Class-sched trans	132	10
B1AAAC	Tape	I	Monthly	100	40
C1CAAC	Disk	I/O	Edit table file	506	2
A1BAAC	Tape	O	Sorted CS trans	132	10
B1AAAC	Tape	O	SSF	100	40
SORTWK1-5	Disk	I/O	Sortwork File	132	12
P1C					
A1BAAC	Tape	I	Sorted CS trans	132	10
C1CAAC	Disk	I/O	Edit table file	506	2
A1CAAC	Tape	O	Edited trans	286	8

Table C-1 Continued.

File Name	Media	Input/ Output	Description	Logical Record Length	Blocking Factor
P1G					
A1CAAC	Tape	I	Edited trans	286	8
R1GAAC	Tape	I/O	Recycle trans	286	8
C1CAAC	Disk	I/O	Edit table	506	2
SORTWK1-6	Disk	I/O			
*1GAAC	Disk	O	A	280	6
where * = A,B,C,E,			B	285	8
F,G,I,J,K,M,N,Q,R			C	285	8
			E	285	8
			F	80	20
			G	125	25
			I	84	20
			J	90	25
			K	280	6
			M	80	20
			N	80	20
			Q	84	41
			R	286	8

APPENDIX D

EXAMPLES OF IAPS INPUT TABLES

This Appendix contains a complete listing of the Application File Table (Figure D-2) and System File Table (Figure D-3) for the first four jobsteps of SIDPERS. It also contains a partial listing of the Application Processing Table (Figure D-4), namely that portion which represents the first two jobsteps of SIDPERS (P1A and P1B). For the reader's convenience, in Figure D-1, we give an explanation of the headings for the Application and System File Tables.

Application File Table

FILE - a user given unique identifier for each application file

LRECL - Logical record length of application file

BLKSZ - Blocksize

#RECS - Number of logical records to be processed

System File Table

FILE - Same as above, to be used as a cross reference between the two file tables

VOLUME - Physical unit type (D for disk, T for tape, C for console) and unit number

LRECL - Logical record length from system's point of view

BLKSZ - Physical blocksize

#RECS - Number of records on file

%PE - Percentage of records on primary extent

#SE - Number of secondary extents

K/U - Placement known (K) or unknown (U)

TYPE - Primary extent (P), index extent (I), overflow (O), or VTOC (V), for disk files only

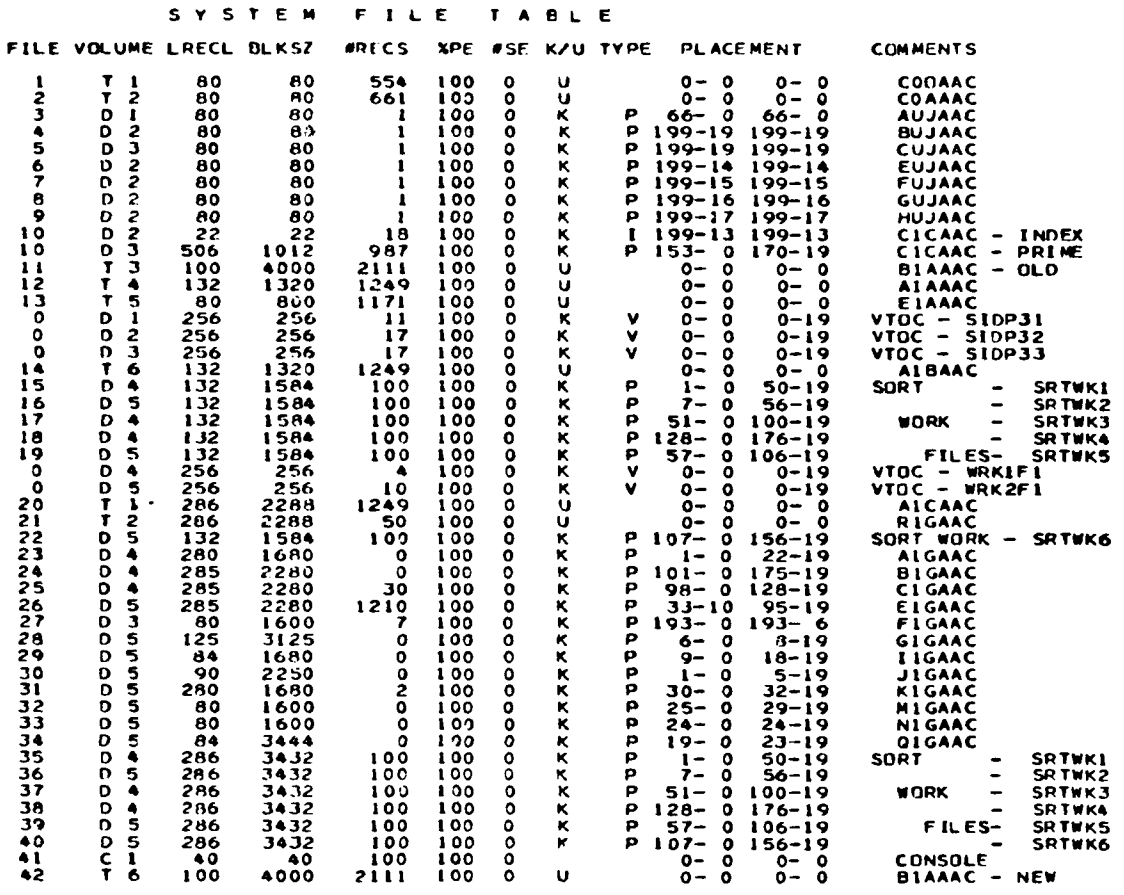
PLACEMENT - Actual placement of disk file, if known, given in low cylinder - low track address to high cylinder - high track address

Figure D-1. Explanation of Headings in Figures D-2 and D-3

F I L E T A B L E

FILE	LRECL	BLKSZ	#RECS	COMMENTS		
1	80	80	554	INPUT TO PIAAACA	- C00AAC	- OPTIONAL CARD INPUT
2	80	80	661		- C0AAAC	- INPUT TRANSACTIONS
3	80	80	1		- AUJAAC	- O P E N E D
4	80	80	1		- BUJAAC	-
5	80	80	1		- CUJAAC	-
6	80	80	1		- EUJAAC	- R E A D
7	80	80	1		- FUJAAC	-
8	80	80	1		- GUJAAC	-
9	80	80	1		- HUJAAC	- C L O S E D
10	506	506	987	INPUT & OUTPUT	- C1CAAC	- COST TABLE FILE
11	100	100	2111		- B1AAAC	- OLD STACKER FILE
12	132	132	1249	OUTPUT PIAAACA	- A1AAAC	- CLASS SCHED TRANS
13	80	80	1171		- E1AAAC	- SIDPERS TRANS HIST
14	132	132	1249	OUTPUT PIBAACS	- A1BAAC	- SORTED CS TRANS
15	132	132	100	S O R T	- SRTWK1	-
16	132	132	100		- SRTWK2	-
17	132	132	100	W O R K	- SRTWK3	-
18	132	132	100		- SRTWK4	-
19	132	132	100	F I L E S	- SRTWK5	-
20	286	286	1249	OUTPUT PICAACA	- A1CAAC	
21	296	286	50	IN & OUTPUT FOR	- R1GAAC	-
22	506	506	100	PIGAACS	- SRTWK6	- SORT WORK FILE
23	280	280	0	OUTPUT PIGAACS	- A1GAAC	-
24	285	285	0		- B1GAAC	-
25	285	285	30		- C1GAAC	-
26	285	285	1210		- E1GAAC	-
27	80	80	7		- F1GAAC	-
28	125	125	0		- G1GAAC	-
29	84	84	0		- I1GAAC	-
30	90	90	0		- J1GAAC	-
31	280	280	2		- K1GAAC	-
32	80	80	0		- M1GAAC	-
33	80	80	0		- N1GAAC	-
34	84	84	0		- Q1GAAC	-
35	286	286	100	S O R T	- SRTWK1	-
36	286	286	100		- SRTWK2	-
37	286	286	100	W O R K	- SRTWK3	-
38	286	286	100		- SRTWK4	-
39	286	286	100	F I L E S	- SRTWK5	-
40	286	286	100		- SRTWK6	-
41	40	40	100	CONSOLE		
42	100	100	2111		- B1AAAC	- NEW STACKER FILE

Figure D-2. Application File Table for SIDPERS



SYSTEM FILE TABLE

FILE	VOLUME	LRECL	BLKSZ	#RECS	%PE	#SE	K/U	TYPE	PLACEMENT		COMMENTS
1	T 1	80	80	554	100	0	U		0- 0	0- 0	C00AAC
2	T 2	80	80	661	100	0	U		0- 0	0- 0	C0AAAC
3	D 1	80	80	1	100	0	K	P	66- 0	66- 0	AUJAAC
4	D 2	80	80	1	100	0	K	P	199-19	199-19	BUJAAC
5	D 3	80	80	1	100	0	K	P	199-19	199-19	CUJAAC
6	D 2	80	80	1	100	0	K	P	199-14	199-14	EUJAAC
7	D 2	80	80	1	100	0	K	P	199-15	199-15	FUJAAC
8	D 2	80	80	1	100	0	K	P	199-16	199-16	GUJAAC
9	D 2	80	80	1	100	0	K	P	199-17	199-17	HUJAAC
10	D 2	22	22	18	100	0	K	I	199-13	199-13	C1CAAC - INDEX
10	D 3	506	1012	987	100	0	K	P	153- 0	170-19	C1CAAC - PRIME
11	T 3	100	4000	2111	100	0	U		0- 0	0- 0	B1AAAC - OLD
12	T 4	132	1320	1249	100	0	U		0- 0	0- 0	A1AAAC
13	T 5	80	800	1171	100	0	U		0- 0	0- 0	E1AAAC
0	D 1	256	256	11	100	0	K	V	0- 0	0-19	VTOC - SIDP31
0	D 2	256	256	17	100	0	K	V	0- 0	0-19	VTOC - SIDP32
0	D 3	256	256	17	100	0	K	V	0- 0	0-19	VTOC - SIDP33
14	T 6	132	1320	1249	100	0	U		0- 0	0- 0	A1BAAC
15	D 4	132	1584	100	100	0	K	P	1- 0	50-19	SORT - SRTWK1
16	D 5	132	1584	100	100	0	K	P	7- 0	56-19	- SRTWK2
17	D 4	132	1584	100	100	0	K	P	51- 0	100-19	WORK - SRTWK3
18	D 4	132	1584	100	100	0	K	P	128- 0	176-19	- SRTWK4
19	D 5	132	1584	100	100	0	K	P	57- 0	106-19	FILES- SRTWK5
0	D 4	256	256	4	100	0	K	V	0- 0	0-19	VTOC - WRK1F1
0	D 5	256	256	10	100	0	K	V	0- 0	0-19	VTOC - WRK2F1
20	T 1	286	2288	1249	100	0	U		0- 0	0- 0	A1CAAC
21	T 2	286	2288	50	100	0	U		0- 0	0- 0	R1GAAC
22	D 5	132	1584	100	100	0	K	P	107- 0	156-19	SORT WORK - SRTWK6
23	D 4	280	1680	0	100	0	K	P	1- 0	22-19	A1GAAC
24	D 4	285	2280	0	100	0	K	P	101- 0	175-19	B1GAAC
25	D 4	285	2280	30	100	0	K	P	98- 0	128-19	C1GAAC
26	D 5	285	2280	1210	100	0	K	P	33-10	95-19	E1GAAC
27	D 3	80	1600	7	100	0	K	P	193- 0	193- 6	F1GAAC
28	D 5	125	3125	0	100	0	K	P	6- 0	8-19	G1GAAC
29	D 5	84	1680	0	100	0	K	P	9- 0	18-19	I1GAAC
30	D 5	90	2250	0	100	0	K	P	1- 0	5-19	J1GAAC
31	D 5	280	1680	2	100	0	K	P	30- 0	32-19	K1GAAC
32	D 5	80	1600	0	100	0	K	P	25- 0	29-19	M1GAAC
33	D 5	80	1600	0	100	0	K	P	24- 0	24-19	N1GAAC
34	D 5	84	3444	0	100	0	K	P	19- 0	23-19	Q1GAAC
35	D 4	286	3432	100	100	0	K	P	1- 0	50-19	SORT - SRTWK1
36	D 5	286	3432	100	100	0	K	P	7- 0	56-19	- SRTWK2
37	D 4	286	3432	100	100	0	K	P	51- 0	100-19	WORK - SRTWK3
38	D 4	286	3432	100	100	0	K	P	128- 0	176-19	- SRTWK4
39	D 5	286	3432	100	100	0	K	P	57- 0	106-19	FILES- SRTWK5
40	D 5	286	3432	100	100	0	K	P	107- 0	156-19	- SRTWK6
41	C 1	40	40	100	100	0			0- 0	0- 0	CONSOLE
42	T 6	100	4000	2111	100	0	U		0- 0	0- 0	B1AAAC - NEW

Figure D-3. System File Table for SIDPERS

```
      I/O PROCESSING TABLE FOR APPLICATION  1
*
*           S  I  D  P  E  R  S
*
*       JOBSTEP - P1AAAACA
*
*         INPUT ONLY
*           1 CARD/TAPE FILE
*             - C00AAC
*
*           1 TAPE FILE
*             - C0AAAC
*
*           7 DISK FILES
*             - AUJAAC, BUJAAC, CUJAAC, EUJAAC, FUJAAC, GUJAAC, HUJAAC
*
*         INPUT & OUTPUT
*           1 TAPE FILE
*             - B1AAAC
*
*           1 DISK FILE
*             - C1CAAC
*
*         OUTPUT ONLY
*           2 TAPE FILES
*             - A1AAAC, E1AAAC
*
*
*
 EXEC      BEGIN P1AAAACA
*
*            INPUT PROCESSING
*
 I              C00AAC - CARD/TAPE OPTIONAL CARD INPUT
  01 X S 100.
*
 I              C0AAAC - INPUT TRANSACTIONS
  02 Y S 100.
*
*I              AUJAAC
* 03 A S 100.
*I              BUJAAC
```

Figure D-4. Application Processing Table for SIDPERS (Jobsteps P1A and P1B)

```
* 04 B S 100.
*I                CUJAAC
* 05 C S 100.
*I                EUJAAC
* 06 E S 100.
*I                FUJAAC
* 07 F S 100.
*I                GUJAAC
* 08 G S 100.
*I                HUJAAC
* 09 H S 100.
*
 I                C1CAAC - EDIT TABLE FILE
  10 T I .44        (FOUR READS OF FILE ONLY - .44 TIMES 987 = 4)
*
 I                B1AAAC - STACKER FILE
  11 Z S 100.
*
*
 P            CPU PROCESSING
  X       COMPARE    SELECT
*
*
 O           MESSAGE TO CONSOLE
  41 M S 23.
*
*
*
*             OUTPUT PROCESSING
*
 O                C1CAAC - EDIT TABLE FILE - 2 WRITES
  10 U I .22
*
*
 O                A1AAAC - CLASS SCHED TRANS FILE
  12 X S 44.4
*
 O                E1AAAC - SIDPERS TRANSACTION HISTORY
  13 X S 47.4
*
*
 O                A1AAAC - CLASS SCHED TRANS FILE
```

Figure D-4 Continued.

```
  12 Y S 53.0
*
 D                E1AAAC - SIDPERS TRANSACTION HISTORY
  13 Y S 56.5
*
*
 C                B1AAAC - STACKER FILE
  42 Z S 58.6
*
 C                A1AAAC - CLASS SCHED TRANS FILE
  12 Z S 2.49
*
 D                E1AAAC - SIDPERS TRANSACTION HISTORY
  13 Z S 2.65
*
 D                 OPERATOR DELAY (RESPONSE TO CONSOLE, OR TAPE MOUNT)
  10000.  10000.  1.0
*
*
*          END JOBSTEP  P1AAACA
*
 EOP
*
*
*
*       JOBSTEP - P1@AACS
*
*         INPUT ONLY
*           1 TAPE FILE
*             - A1AAAC
*
*           1 DISK FILE
*             - C1CAAC
*
*           1 TAPE FILE (MONTH END ONLY)
*             - B1AAAC
*
*         INPUT & OUTPUT
*           1 TAPE FILE
*             - B1AAAC
*
*         OUTPUT ONLY
```

Figure D-4 Continued.

```
*            1 TAPE FILE
*              - A1BAAC
*
*
*         P1BAACS SORTS FILE A1AAAC & PUTS SORTED OUTPUT INTO A1BAAC.
*         AT MONTH END IT ALSO SORTS B1AAAC.
*
*
*
*
 EXEC      BEGIN P1EAACS
*
*                     BEGIN SORT PHASE
*
 I              C1CAAC - EDIT TABLE FILE - 3 READS
  10 T I .304
*
*
 I              A1AAAC - CLASS SCHED TRANS FILE
  12 X S 35.3
*
 P
  X      COMPARE    SORT
*
 O
  15 Y S 440.
*
 O              MESSAGE TO CONSOLE
  41 M S 13.
*
*
 EOP
*
 I              A1AAAC - CLASS SCHED TRANS FILE
  12 X S 35.3
*
 P
  X      COMPARE    SORT
*
 C
  16 Y S 440.
*
```

Figure D-4 Continued.

```
 EOP
*
 I                A1AAAC - CLASS SCHED TRANS FILE
  12 X S 29.7
*
 P
  X       COMPARE    SORT
*
 O
  17 Y S 370.
*
 ECP
*
*                          END INITIAL SORT PHASE - NOW MERGE
*
*
 I
  15 X S 440.
 I
  16 X S 440.
 I
  17 X S 370.
*
 P
  X       MERGE
*
 O                A1BAAC - SORTED CLASS SCHED TRANS FILE
  14 X S 35.3
 O
  14 X S 35.3
 O
  14 X S 29.7
*
 C                  OPERATOR DELAY (RESPONSE TO CONSOLE OR TAPE MOUNT)
  10000.  10000.  1.0
*
*
*
*
*           END JOBSTEP  P1BAACS
*
 EOP
*
```

Figure D-4 Continued.

APPENDIX E

THE MODEL LIBRARY

What follows is a complete list of the currently existing members of the model library, followed by a brief discussion of those members which the User would have to be familiar with in order to run models.

$GADRU	-	IPSS Get Address routine, modified for IAPS methodology
BUFMR	-	Buffer manager
CHPGM30	-	Channel program for IBM 360/30 hardware
CHPGM47	-	Channel program for Honeywell Model 47 hardware
HDWM30	-	Hardware specifications for IBM 360/30, all variants
HDWM47	-	Hardware specifications for Honeywell Model 47, all variants
M30A2	-	Hardware configuration A2 (See Table 5-3)
M30A3	-	Hardware configuration A3
M30A4	-	Hardware configuration A4
M47B2	-	Hardware configuration B2
RDGP	-	Reads Application Processing Table and prepare it for processing
RSUF	-	Reads System and Application File Tables, and set up Index Tables
SERVICES	-	Application program processing
SIDIOPT	-	SIDPERS Application Processing Table (for the first four jobsteps of SIDPERS)
SIDSFT	-	System File Table for SIDPERS
SIDSFTB2	-	System File Table for SIDPERS for configuration B2
SIDSFT2T	-	System File Table for SIDPERS with 2 tape files (other tape files transferred to disk)
SIDVFT	-	Application File Table for SIDPERS
SID30A1	-	Hardware configuration A1
SID47B1	-	Hardware configuration B1
STORAGE	-	Generalized database description

The following members form the core of the IAPS methodology and would be in every model; thus they need not concern the User.

$GADRU
BUFMR
RDGP
RSUF
SERVICES
STORAGE

The next group of members define the hardware configuration and thus require a User choice. The brackets indicate that a choice of one and only one must be made.

STD30A1
M30A2
M30A3
M30A4
STD47B1
M47B2

If one of the first four configurations is chosen, then hardware specifications will come from HWDM30 and CHPGM30; otherwise, HDWM47 and CHPGM47 will be used.

The second and final decision made by the User before submitting a run involves the workload, or loading. At present, there is only one Application Processing Table, SIDIOPT, which models the first four jobsteps of SIDPERS, and there is only one Application File Table, SIDUFT, which specifies the file characteristics of SIDPERS files from the Cobol programmer's point of view. However, there are 3 choices for System File Table, namely SIDSFT, SIDSFTB2, and SIDSFT2T. The first choice, SIDSFT, places all files on disk and tape units exactly as current Army practice, while the latter two offer slight variations on file placement to accommodate different hardware configurations.

In summary, to run models the User would have to make two choices, one on the hardware configuration desired and the other on the desired loading.

APPENDIX F

GUIDE TO PREPARING IAPS INPUT

This appendix contains detailed formatting information for the three input tables required to use the IAPS methodology. The three tables are the Application File Table, the System File Table, and the Application Processing Table. Examples of these tables for SIDPERS are in Appendix D.

Application File Table Format

The application file table is a description of each file from the application programmer's point of view.

Column	Code	Explanation
1-2	Any number from 01 to 50, no two identical	A unique file identifier
3	Blank	
4-7	Logical record length (in bytes)	
8	Blank	
9-14	Blocksize (in bytes)	
15	Blank	
16-21	Number of logical records in file	
22-72	Modeler comments	

System File Table Format

The system file table is a description of each file from the hardware point of view.

Column	Code	Explanation
1-2	Any number from 01 to 50	The file identifier from the application file table.
	0	Any system file not directly referenced by a user's program
3	Blank	
4	T, D, or C	T - Tape D - Disk C - console
5	Blank	
6	Device unit number (01 to 50)	
7	Blank	
8-11	Logical record length	
12	Blank	
13-18	Blocksize	
19	Blank	
20-25	Number of logical records in file	
26	Blank	
27-29	Percent of records in primary extent	
30	Blank	

System File Table Format (continued)

Column	Code	Explanation
31-32	Number of secondary extents	
33	Blank	
34	K or U	K - known placement U - unknown placement
35	Blank	
36	I, P, O, V or Blank	I - index extent P - prime extent O - overflow extent V - VTOC Blank - prime extent (for tape files)
37	Blank	
38-40	Low cylinder address (LCA)	The pair LCA-LTA gives the beginning address of the file on disk
41	Blank	
42	Low track address (LTA)	
43	Blank	
44-46	High cylinder address (HCA)	The pair HCA-HTA gives the ending address of the extent allocated to the file
47	Blank	
48-49	High track address (HTA)	
50-72	Modeler comments	

Application Processing Table Format

The Application Processing Table describes the processing performed by application systems. This table allows the specification of I/O activities, CPU processing and delays. Table entries are easily grouped into identifiable packets of real world activities (job steps and jobs), and allow for user comments.

a) Comment cards may appear anywhere within the application processing input except between an I, O, or D processing specification card and its associated definition card.

Column	Code	Explanation
1	*	Card is ignored by processor
2-72	User comments	

b) Delimiter cards mark the beginning and end of processing and execution groups.

Column	Code	Explanation
1	Blank	
2-6	EXEC	Begin an execution group
2-5	EOP	End processing group
7-72	User comments	

c) Processing specification cards model the input, output, and processing activities within each processing group.

Column	Code	Explanation
1	Blank	
2	I	Input file Zero or more occurences Must proceed all "P" and "O" cards within a processing group
	P	Processing option One occurrence Must proceed all "O" cards within a processing group
	O	Output file Zero or more occurrences Must follow "P" card
	D	Delay option No more than two occurrences Must be the first and/or the last processing specification card in a processing group
3-72	User comments	

d) An I/O definition card must follow each I or O specification card.

Column	Code	Explanation
1-2	Blank	
3-4	Any integer from 01 to 99	Application file number
5	Blank	

d) continued.

Column	Code	Explanation
6	Blank	Nonconcurrent activity
	Any non-blank alphanumeric character	Concurrent activity
7	Blank	
8	S	Sequential access
	R	Random access
	I	ISAM file
	V	VSAM file
9	Blank	
10-12	Any integer from 0 to 100	Percent of records processed
13-72	User Comments	

e) At least one P definition card must follow the P specification card.

Column	Code	Explanation
1-2	Blank	
3	Blank	Processing not concurrent with any I/O
	Non-blank	Processing concurrent with associated I/O
4-9	Blank	
10-19	Any of the codes: SORT MERGE COMPUTE EDIT UPDATE SELECT REPORT USEREXIT	Defines processing activity (Each code must start in col 10)

e) (continued)

Column	Code	Explanation
20-29		Coded as col 10-19
30-39		
40-49		First blank field terminates card
50-59		
60-69		

f) Exactly one D definition card must follow each D specification card.

Column	Code	Explanation
1-2	Blank	
3-9	Any nonnegative decimal number	Estimated minimum delay (All 3 numbers should have a *decimal point*)
10	Blank	
11-17	Any nonnegative decimal number	Estimated maximum delay
18	Blank	
19-25	Any number between 0.0 and 1.0	Probability of a positive delay

APPENDIX G

GRASP ACCOUNTING DATA

The following is a partial listing of the measurement data provided by the GRASP accounting package. This list is included in this report to indicate the wide variety and usefulness of GRASP step accounting data to computer simulation projects.

Table G-1. GRASP Accounting Data

Partition	- identifies the partition in which the job executed
Time-on	- time of day the job began
Time-off	- time at which the job ended
Duration	- time the job occupied the partition
Non-MPS duration	- time the job would have run without interference
Interference duration	- time this job was interfered with by multiprogramming activity
CPU time	- time spent by this job executing CPU instructions
Operator duration	- time spent by this job in wait states of 3 seconds or longer
I/O wait time	- time spent waiting for data transfer to complete
Phase loads	- total fetches or loads performed by this job
Time waiting for LTA	- total time waiting for access to the transient area
Time using LTA	- total time the LTA was used by this job
Lines spooled	- number of print lines produced by this job
Cards spooled in	- number of input cards spooled for this job
Cards spooled out	- number of cards punched and spooled for this job
Start I/O counts	- the number of input or output requests issued for each symbolic logical unit used by this job

Table G-1 Continued.

I/O device usage time	- the total "device busy" time accrued on each I/O device used by this job
SYSRES usage time	- time spent by the job reading or writing on the SYSRES device
CPU utilization	- total time the CPU was active for any partition/purpose during the execution of this job
Channel activity	- total time each channel was "active"
CPU channel overlap	- overlap between CPU and channel activity
Core used	- total size of the program as loaded into storage